UNIVERSITY OF MALAYA

CONSTRUCTION OF VIRTUAL ENVIRONMENTS THROUGH INTEGRATION OF THREE DIMENSIONAL (3D) OBJECTS

SESSION 2003/2004
WXES 3182 PROJEK ILMIAH TAHAP AKHIR II
MOHD FAUZI B. MUSTAFA
WEK 010170

SUPERVISOR: CIK SU MOON TING
MODERATOR: PUAN ZARINAH MOHD. KASIRUN

DATE OF SUBMISSION: 11 October 2004

Abstract

Annually, all the undergraduates of Faculty of Computer Science and Information Technology, University of Malaya are required to prepare a final project as part of the degree requirement. It is hoped that the student will be apply concepts learned in the university into the production of the project. For my final project, I have chosen to do a study and design of a three-dimensional (3D) integration tool which able to integrate 3D objects into the virtual environment.

This is a group project that develops a tool which is user-friendly and suitable for user who always desire to integrate 3D objects in a single virtual environment and to perform transformation to 3D objects accordingly to their needs. The user will be able to integrate 3D objects from the provided 3D objects by the tool or import it from other sources. After integration the user will be able to perform transformation such as rotation, scaling and rotation to the 3D objects within the virtual environment. . With this function, the user will be able to arrange the location, size and view of their 3D objects according to their needs and desires.

The Waterfall Model with Prototyping approach was selected for the development process because the strength of both the Waterfall Model and Prototyping can be combined in a single project and reduces the risk involved. The programming languages that used to develop the tool are Microsoft Visual C++, Visual Basic.net, OpenGL and Microsoft Windows XP has been chosen as the operating system.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God for guiding me and giving me the patience and concentration in doing this project. In the process of doing this project, I was facing challenges that sometimes seem to be difficult at first, but with perseverance and focus, I was able to complete the thesis smoothly on time.

I would like to take this opportunity to thank a few people, Ms. Su Moon Ting as my supervisor and lecturer who helped me a lot and guide me through the process of doing this project. Without your knowledge and consultation, there is no way this project will work.

I would like to thank also to Mrs. Zarinah Mohd. Kasirun, my moderator for this project, who had given me constructive advise so that this project will be even more challenging and produce a better output.

I want to express my gratitude to my partner, Ms. Nurhairos Hamdan who have helped me and give me support and assistance to me throughout the project. Also not forgetting my family at home, my parents who always pray for my success in whatever thing I do. This will always be the driving force behind me.

Finally, I would like to dedicate this project to my friends in Kelana Jaya.

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

List Of Figures **Page**

List Of Tables **Page**

CHAPTER ONE: INTRODUCTION

Chapter 1

Introduction

The extensive growth of graphics technology brings human into a new era. The war between static two-dimension (2D) images and dynamic three-dimension (3D) virtual spaces seems like inevitable. Obviously, the success takes the side of 3D. The used of 3D in representing information is far better than 2D. For example, 3D can exactly point out the structure of human DNA but 2D cannot. In the 3D virtual spaces, people could communicate and interact with the virtual world. This is the contribution of internet.

Virtual reality (VR) is another innovative media technology that has become part of the multimedia pattern in the coming years. It allows the user to feel the experience fantasy situations that are generated by computers which supported by the interactive software. As an information tools, it could add new dimensions to understanding of the world.

The applications of virtual 3D environments are not limited to the entertainment industry; they have permeated into industrial design, biotechnology, architecture, urban planning, defense, development of certain tools. At the heart of such a wide variety of application lie core software components to render 3D content efficiently and realistically [The Amazing World of Virtual 3D Environments].

Recent years have been increasing use of large 3D models in various application domains such as engineering design and manufacture, architecture, medical imaging, Graphical Imaging System (GIS), art restoration studies and heritage site documentation. In addition to geometry information they often have rich textures, associated non-visual

information and behavior such as dynamically changing geometries, response to integration and more.

1.1 Project Overview

"Construction of Virtual Environment through Integration of 3D Objects" can be described as a tool to integrate 3D objects in separate files into a single virtual environment. This mean that integration is the main aspect for this project and it suit for user that needs to integrate their own objects such furniture and others.

The project is to develop a 3D integration tool which able to integrate 3D objects into a single virtual environment. The tool will allow user to integrate 3D objects which can be imported from other sources or using the provided by this tool. During the integration, this tool able to provide transformation functions such as rotation, scaling and rotation to the 3D objects. With this function, this 3D integration tool will allow the user to modify the current location, size and rotation of their 3D objects according to their needs and desires. After the integration, this tool allows the user to save the virtual environment.

1.2 Project Objective

The objectives of the project are listed below:

- To create a tool that can integrate separated 3D object files into a single virtual environment. This is the part of this project which will be developed by me.
- The tool should allow functions such as translation, rotation and scaling to be applied during the integration of the 3D objects. This is transformation function of this project which will be developed by me.
- To create a tool that should be able to save files after integration of the 3D objects. This part of the project will be developed by both of the group members together.
- Allow user to get and retrieve various kind of 3D objects files. This part of the project will be developed by both of the group members together.
- To achieve paperless environment simultaneously reducing the amount of design draft.
- To reduce time effort and effort for designing due to level of human interaction found in manual design process.

1.3 Project Scopes

This project is a tool that put extra focus on function like scaling, rotation and translation to an object. Means that this tool is not a kind that develops or modeling 3D objects like other modeling tools like 3DStudio Max or Lightwave3D. There will be provided 3D objects in this tool. Users will be provided 3D objects by this tool or they also can import 3D objects from other sources.

Since this project is more focus on integration in a virtual environment, so others interactive effect like audio clip is not included in this tool. But basic function like save file is available in this tool.

1.4 Target Users

- 3D Modeler or 3D Developer

Web3d developers that often use 3D tools can use this software to integrate their variety objects in different environment. Other users like 3D game developer also can modify 3D objects with basic features like translation and rotation with this modeling tool.

1.5 Project Schedule

In order to timely achieve project objectives, a project schedule is planned rationally to manage the time taken for each task. The schedule has been divided into two phases. There are 2 Gantt charts for the first and second phase of this project. The first phase will be during the period March-May 2004, and the second phase will be June-October 2004

Activities/Month	March 04	April 04	May 04
Title Selection/Discussion with lecturer			
Research/Reading/Consulting			
Final Discussion/Analysis			
Final Proposal and Viva			
Writing Report/Submission			

Table 1.1: Gantt Chart Phase 1

Activities/Month	June 04	July 04	August 04	Sep 04	Oct 04
Software Development					
Testing/Simulating					
Documentation					
Viva					
Writing Report/Finalizing					

Table 1.2: Gantt Chart Phase 2

1.6 Expected Outcome of Project

- Able to help 3D modeler to create a satisfied virtual environment based on their desire.
- Has a user friendly interfaces that help users to grasp all the features in the software easily without any difficult.
- Able to help 3D modeler to get and retrieve various 3D objects and combine in one virtual environment that can be modifying by themselves.

1.7 Summary

In Chapter One, the project objective, project scope, project unique features, target users, project schedule and expected outcome of this project have been clearly clarified. The next chapter literature review will carried out whereby current systems are surveyed to better understand how it is implemented, together with comparisons between different operating system platforms, development tools and others.

CHAPTER TWO: LITERATION REVIEW

Chapter 2: Literature Review

2.1 Purpose of Literature Review

Review of literature is a background study about the knowledge and information gained to develop this project. Its' purpose is to get a better understanding on the development tools that can be used to develop a project and also to get a better knowledge on the development methodologies used while developing a project. Besides that, review of literature also enables the developers to do comparison on the past-developed projects and study the strength and weakness of it. It will also give an overview of how to improve the weakness and fulfill the requirements needed.

2.2 Information Gathering Method

In developing a system, it is important to identify the system requirement. In order to identify them, a lot of information is needed. A few techniques have been used to find out what the system needs and users really want. The requirement elicitation takes quite a long time. This is due to several techniques need to be applied in order to get a complete requirement. These include;

2.2.1 Internet Research

Internet is used as the main resource for referring any ambiguities that arise during the entire development period. By analyzing in the similar system documentation has made a big help in giving ideas on the features, functionality as well as the design of the web-based system. Besides that, online tutorials regarding programming language can also be obtained through surfing the Internet.

2.2.2 Printed Media

A lot of published literatures have been read in order to gather information of the users' needs, system development needs and technical issues of the proposed system. All these can be categorized into the printed material (especially books and journal) and non-printed material such as electronic document and the likes. Through reading, ideas are managed to get from books, magazines and journal. Much useful information has been found from PC Magazine and In-Tech from local newspaper, Star that provides me with the latest technology in virtual reality and multimedia.

2.2.3 Document Room

Previous seniors' thesis also have been read through in order to gain an overall understanding on how a system was developed, what were the functional and non-functional requirements, and other related data. The general structure of each thesis has also been observed to find out the steps taken in carrying out a thesis.

2.2.4 Observation

The current software of the 3D modeler had been reviewed. Through this technique, the method of searching products and purchase online used in the current system has been observed and defined. Besides, the current system is test to find out the functionality and the problem faced by the system itself. Review has been carried out to see whether it is economic to apply the new system and whether there is enough equipment to develop the new system.

2.2.5 Meetings with the Supervisor

Meetings with my supervisor are carried out at least once a week to discuss about each aspect in the software. This is to ensure that the progress of the project will lead to its objective.

2.3 Three Dimensional (3D)

3D is common short term for "three dimensional". Generally these three dimensions are represented as x, y and z co-ordinates. A truly 3D object is also called "volumetric" as it contains not only a surface but an interior [cornell.edu, 2004].

2.4 Virtual Reality

2.4.1 What is virtual?

This refers to something that we can not physically see or touch but exists on the Internet in cyberspace. Virtual used to describe the activities and places that are conducted and replicated on the Internet. Virtual environments are often rendered, rather than photographed and give the illusion of three dimensions (3D) [jvlnet.com, 2004].

2.4.2 What is virtual reality?

Computer generated technology which allows the user to interact with data that gives the appearance of a 3D environment. The user can navigate around a 3D world and interact with objects in that world. It also defines as 3D visual computer stimulation that responds to your input so realistically that you feel you are inside another world. Virtual reality is used in many real life applications, from chemistry to architecture and computer games. Popular product for creating virtual reality effects on personal computer includes Bryce, Extreme 3D, Ray Dream Studio, TrueSpace, 3D Studio Max and Virtual Reality [mtholyoke.edu, 2004]

2.5 Reviewing On Existing System

2.5.1 3D Studio Max 6

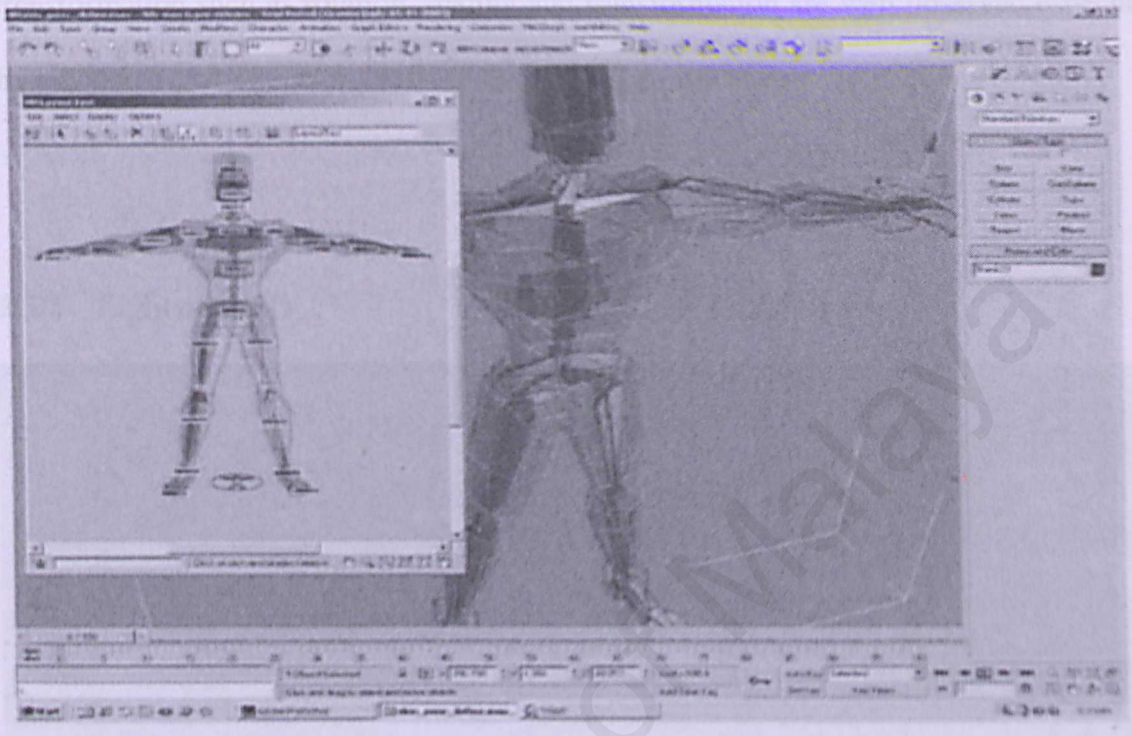


Figure 2.1: 3D Studio Max

3d Studio Max 6, the world's most widely used professional 3D modeling, animation and rendering solution, offers the ultimate professional 3D tools required for creating eye-catching visual effects, cutting-edge games, and distinct design visualizations. 3D Studio Max is not specifically a 3D integration tool. It is a 3D modeling, animation and rendering software.

3D Studio Max offers remarkable possibilities for the modeler and animator and is especially good for the individual or company who wants a bit of everything. With all the improvements in this version, it definitely offers good value for the money

3D Studio Max offers feature can be used by architects, among others, to show as closely as possible what a design would look like when implemented. Features like transformation such as rotation, scaling and transformation are well observed especially by me which I will develop these function for this project.

2.5.2 Lightwave 3D

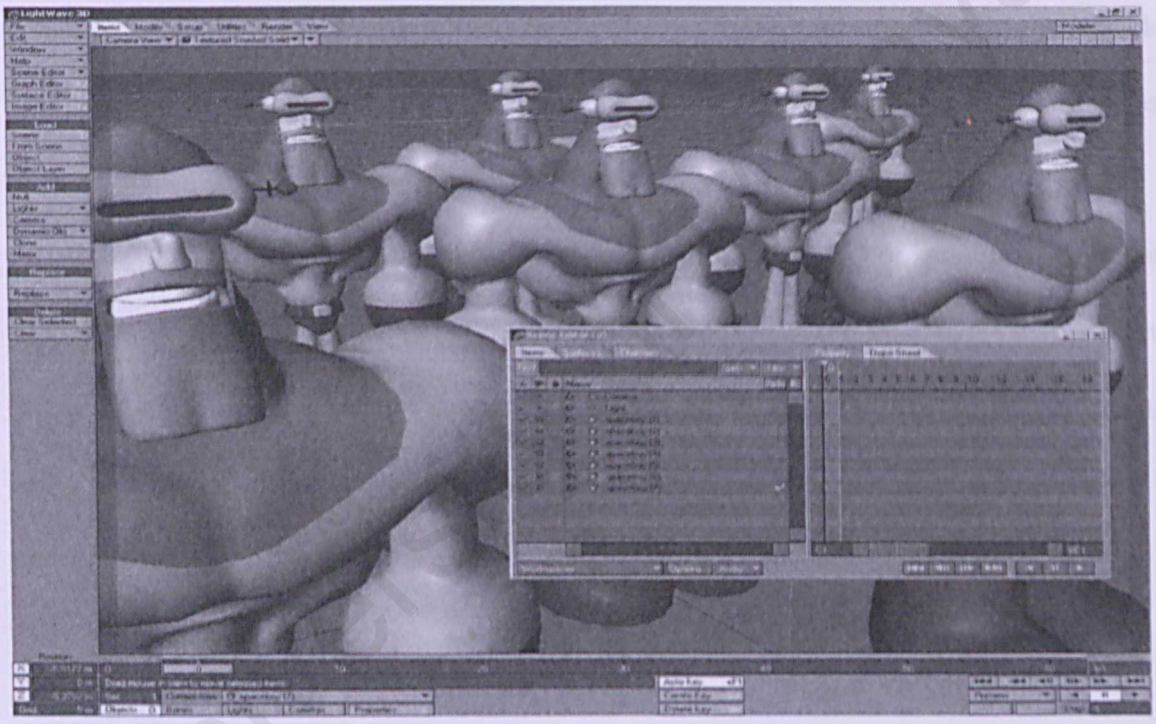


Figure 2.2: LightWave 3D

LightWave 3D is the most complete and flexible software solution for 3D graphics and animation. Thus it's interface is much more sparse in appearance, and may be considered "Un-Mac" like to some. However, things have changed a great deal with the latest version and many of the Interface colors are now customizable. The LightWave Suite is actually a paired set of two separate applications; Modeler for

creating Three Dimensional objects & basic Surface attributes and Layout for Animating or moving those objects & Surfaces. Modeler is comprised of a single application window which is scalable and divided into Tabs or Modes at the top left, Functions or Tools that correspond to the Tabs on the left panel, and 4 basic divisions of Side, Top, Rear, & Perspective views for the rest of the window. There are no "fancy" icons & everything is labeled with text. There is also a 3rd application included in the package known as Screamer Net which allows one to send sets of animations to a multitude of computers on a network & patch them all together to make one seamless animation file. This works without even having LightWave on all the computers! Very handy for those of you with multiple G3s & G4s linked via Ethernet hub.

With LightWave 3D, user able to import and integrate 3D objects into a template virtual environment or a new environment. In addition, user also can perform save function to the single environment after modeling its 3D objects. All such functions are being well studied for this project and other functions are less related to this project.

2.5.3 Vizx3D

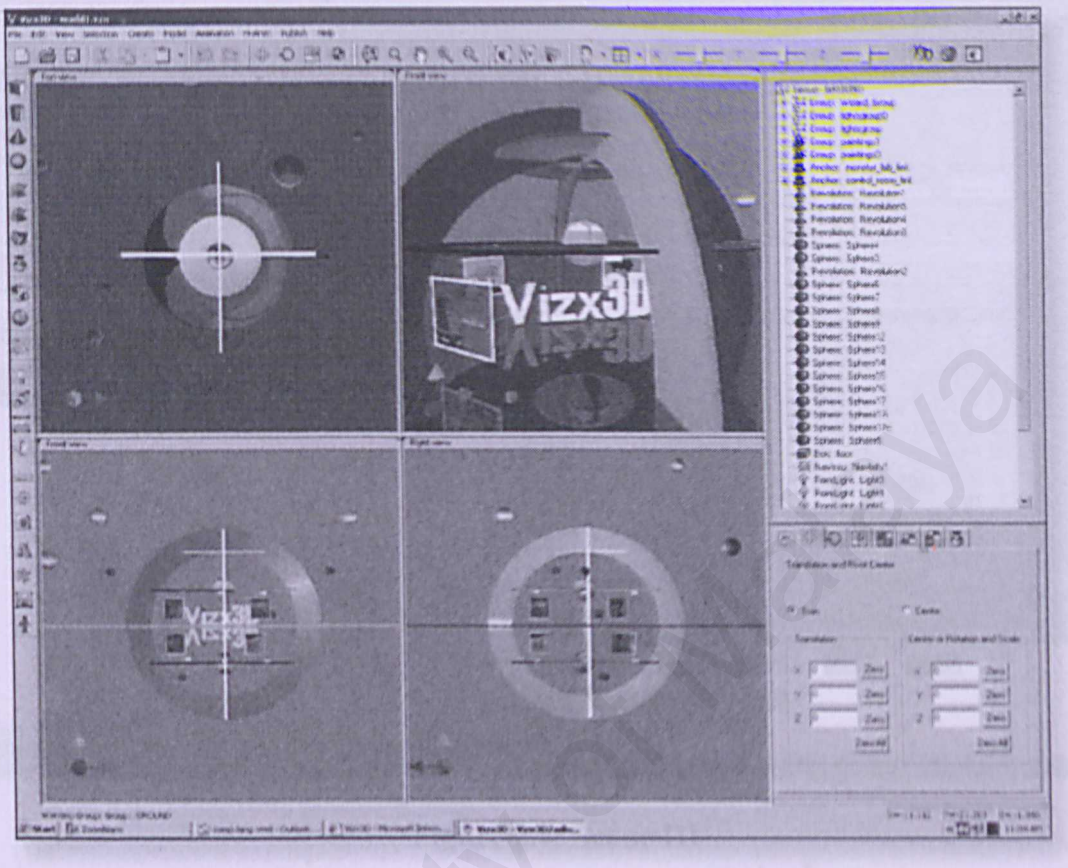


Figure 2.3: Vizx3D

This 3D modeling and animating application creates X3D VRML97 files. To view the files, you will need 3D plug-in for your Web browser. Features include H-Anim 2001 single-mesh avatar support, an improved IFS/NURBS surface editor, subdivided surfaces and multi-texture functions. It is having been my concern about Vizx3D which offers 3D integration and transformation features as it will be develop by me for this project. There are provided objects in this tool that concern for this project and it is designed well with others user interface.

2.5.4 Scene III

Figure 2.4: Scene III

Scene III makes it easy to seamlessly integrate 2D and 3D animation. There are distinct benefits to be gained by integrating 2D and 3D. For example, some things are quicker, easier and cheaper to do in 3D such as material properties, rotating animation, props and backgrounds which are drawn as 3D and rendered as 2D. Other things are quicker, easier and cheaper to do in 2D, such as character animation. Scene III lets you use both methods to create 2D animation.

With Scene III, the 3D data is shown in 3 windows. There is 3D World Graph; shows the elements that make up the 3D model in a tree format. 3D World View will show the geometry of the 3D data and includes a view of elements such as lights and camera and F Curve Editor will show a graphical representation of the animation data. 3D animation can be modified in this window, allowing animators to fine-tune their 3D

models and save adjustments to position and timing. Using the various views of 3D data, Scene III allows users to make minor changes in individual elements of the 3D model - such as translation, scaling and rotation of 3D model data, and adding lights and cameras - without having to go back to the original 3D animation package.

2.6 Development Tools Consideration

2.6.1 Programming Tools

2.6.1.1 Microsoft Visual Basic.net

Visual Basic .NET, the next generation of Visual Basic, is designed to be the easiest and most productive tool for creating .NET applications, including Windows applications, Web Services, and Web applications. While providing the traditional ease-of-use of Visual Basic development, Visual Basic .NET also allows optional use of new language features. Inheritance, method overloading, structured exception handling, and free threading all make Visual Basic a powerful object-oriented programming language. Visual Basic .NET fully integrates with the .NET Framework and the Common Language Runtime, which together provide language interoperability, simplified deployment, enhanced security, and improved versioning support [Microsoft.com, 2004].

Microsoft Visual Basic.net support full object-oriented construction to enable more components, reusable code. Visual Basic language features include inheritance, encapsulation and polymorphism.

Microsoft Visual Basic.net is capable to bind with others Visual Studio applications such Visual C++ and its very important features for this project. In addition, it also able to develop and designing and building application with graphical user interface.

2.6.1.2 Java

Java is a powerful computer programming language that is fun to use for novices while simultaneously being appropriate for experienced programmers building substantial information systems. Java is certain to become the language of choice in the new millennium for implementing Internet-based and Intranet-based applications and any other software for devices that communicate over a network.

Java is a full-featured computer language that incorporates the best of modern thinking about Object Oriented (OO) programming. Java is simpler and more robust than other computer languages and combines features, which make it ideal for programs, which must deal with networks. The designers of Java emphasized securities, ease of programming and independence from any particular hardware. These features brought Java near instant acclaim in the programming word and meteoric rice in public consciousness [Sun MicroSystem 7,2002; Sun MicroSystem 8, 2002].

2.6.1.3 Visual C++

C++ is a superset of C develops by Bjorne Stroustrup at Bell Laboratory. C++ provides a number of features that "spruce up" the C language. More important, it provides capabilities for object oriented programming (OOP).

Object is essential reusable software component that model item in the real world [Harvey and Paul Deitel, 2000]. Object oriented programming is one of the methods which are able to allow a programmer to define a group of variables and function, which known as class. The object concept allow programmer to reuse this set of variables and functions in any part of the program.

Beside classes, C++ does have features that C and Visual Basic do not really have, for example C++ templates, operator overloading, inheritance, virtual function and polymorphism. With this features it allow a C++ programmer more freedom in programming and more powerful. In addition, it also allowed C++ to integrate with application programming interface (API) such as OpenGL which make it very important features for this project.

2.6.2 Hardware Application Programming Interface (API)

Application Programming Interface (API) is a set of routines, protocols, and tools for building software applications. A good API makes it easier to develop a program by providing all the building blocks. A programmer puts the blocks together. Although APIs are designed for programmers, they are ultimately good for users because they guarantee that all programs using a common API will have similar interfaces. This makes it easier for users to learn new programs. A good API makes it easier to develop a program by providing all the building blocks.

2.6.2.1 OpenGL

OpenGL is popular among developers in many different industries. It provides a great deal of functionality and has proven to be stable Application Programming Interface (API) over 10 years of its existence.

OpenGL is at its core, a state machine which control how primitives are processed and rendered. It uses a procedural model (i.e. function) to modify the state machine and pass data to it. Due to the high effiency of this mechanism, code is usually short n easy to understand. This makes it easier to debug. Although OpenGL itself is

procedural, it can be used in linear, modular or OOP fashion equally easily; the choice is depend on the programmer's. OpenGL conceals a lot of detail about specific hardware devices such from programmer. Most basic operations are very easy to do. Rendering is done between the calls glBegin and glEnd.

All OpenGL functions use a naming convention that looks like glFuntionName [argtypes]. An example of an OpenGL style function would be glVertex3f. "gl" tells you that is an OpenGL call. "Vertex" tells u that you are specifying a vertex. "3" tells you that the vertex is being specified in 3 dimensions, and "f" means the arguments are floats. (Note that the number isn't present in most functions, but when it is present it refers to how many component or dimensions the function specifies). To help portability, OpenGL define custom types, all prefixed by "GL", which maybe followed by a "u" for unsigned values, and then a name indicating the type, such a short, long, float, etc. An example type is GLuint.

One final thing to note is that OpenGL syntax for C is much more intuitive and simple than the C syntax for D3D (Direct3D) [Promit Roy, Direct3D vs. OpenGL]. OpenGL is very portable, it run for nearly every platform in existence even runs on Windows NT 20002. It also has a wide range of features, both in its one and through extensions. Its extension feature allows it to stay immediately current with new hardware features; the features available in OpenGL represent a wide range of interest thus make it useful in many different application.

OpenGL also has some weaknesses. OpenGL extensions are powerful but they do make code messy, very much so at times. They also make it confusing with any compiler that doesn't offer preference tracking C (browse file). The worst part is many newer extensions are completely card or vendor specific (Only support certain graphics

card only). Although useful for testing a graphic's card's abilities, vendor- specific extensions are not frequently used by commercial applications.

2.6.2.2 Virtual Reality Modeling Language (VRML)

VRML or Virtual Reality Modeling Language is a standard language for three-dimensional objects, scenes or 'worlds' delivered across the internet. Virtual Reality Modelling Language (VRML) is a language for describing three dimensional (3D) image sequence and possible user interactions to go with them. By using VRML, users can build a sequence of visual images into web settings with which a user can interact by viewing, moving, rotating and otherwise interacting with an apparently 3D scene. For example, users can view a room and use controls to move the room and experience it by walking through it in real space.

In the case of VRML worlds, these can be animated and allow for the user to create an avatar or symbolic representation of him/herself in order to actively move about and interact with other users/avatars in a 3D space such as a 'room' or 'gallery'. VRML multi-user worlds, a standard which is still developing through experiments going on now, adds visual information and 3D objects to MUDs (Multi-User Dimension or Multi-User Dungeons) which are currently tect-based.

VRML features are navigation, viewpoint, model, material sound, textures, lighting, special nodes, performance, collisions, animation, sensors, scripting, routers, compact, modular and extensible. These features are well concerned which closely related to the related functions such as transformation for this project.

2.6.2.3 Java 3D

Java 3D API is an application programming interface used for writing three-dimensional graphics applications and applets. It gives developers high-level constructs for creating and manipulating 3D geometry and for constructing the structures used in rendering that geometry. Application developers can describe very large virtual worlds using these constructs, which provide Java 3D with enough information to render these worlds efficiently.

Java 3D delivers Java's "write once, run anywhere" benefit to developers of 3D graphics applications. Java 3D is part of the JavaMedia suite of APIs, making it available on a wide range of platforms. It also integrates well with the Internet because applications and applets written using the Java 3D API have access to the entire set of Java classes.

The Java 3D API draws its ideas from existing graphics APIs and from new technologies. Java 3D's low-level graphics constructs synthesize the best ideas found in low-level APIs such as Direct3D, OpenGL, QuickDraw3D, and XGL. Similarly, its higher-level constructs synthesize the best ideas found in several scene graph-based systems. Java 3D introduces some concepts not commonly considered part of the graphics environment, such as 3D spatial sound. Java 3D's sound capabilities help to provide a more immersive experience for the user.

2.6.2.4 DirectX

DirectX was considered to be pretty bad compared to OpenGL for the past release. Recent advancement in the API, however have made it very powerful and stable. In fact, DirectX is well organized as the standard for graphic on window platform. Microsoft works very closely with graphics hardware companies to ensure that any new features that they introduce will be supported in Direct 3D. Often, Direct 3D supports features before "card" do. DirectX is based on the COM object model which mean, programmer creates private classes to those rather than just calling functions that don't clearly show associations.

Directx3D does something extremely well. Two of its major strengths, added with DirectX 8, are Programmable Pixel and Vertex Shades. In surface they allow programmer to replace portions of the rendering pipeline with custom code shads are written with a sort of graphics language that looks very much like assembly code. On the other hand, only the newest graphics cards (such as GeForce 4 or Radeon 8500) support pixel shades.

Beside than that, DirectX also has other advantages. Direct3D provides functions to enumerate exactly what graphics hardware is available on a system. Direct3D's syntax and structure have evolved to format that many developers believe is more intuitive that previous version making it easier to develop for then it used to be. It's interface well supported in object oriented programming and COM in particular.

DirectX also has some weaknesses. DirectX is still unable to up-to-date every year or so, which is little slow considers have fast the graphic industry moves. They makes up for this somewhat by working closely with graphics vendor and adding support for things aren't yet available. Also, when new features are introduced, Direct3D

offers a standardized way of a accessing them. Direct3D is not an open standard. That mean Microsoft only has the decision on the development of the new release.

2.7 Operating System Consideration

An operating system (sometimes abbreviated as "OS") is the program that, after being initially loaded into the computer by a boot program, manages all the other programs in a computer. The other programs are called applications or application programs. The application programs make use of the operating system by making requests for services through a defined application program interface (API). In addition, users can interact directly with the operating system through a user interface such as a command language or a graphical user interface (GUI) [gslis.utexas.edu, 2004]. For this project Microsoft Windows 2000 and Microsoft Windows XP are reviewed for the project operating system.

2.7.1 Microsoft Windows 2000

The Microsoft® Windows® 2000 Server operating system provides the services you need to put the Internet to work for your business. From publishing basic information about your company to creating a full-blown e-commerce application, starting with Windows 2000 is a great way to build the Internet into your business. Using industry-standard hardware, software, and skills, along with the services and features in Windows 2000, you can readily share information and conduct transactions

involving your employees, customers, and business partners—anywhere in the world—through the Web.

To provide maximum programming flexibility, Windows 2000 uses a component object-based programming model that lets developers use a broad array of Microsoft and third-party development tools to create applications and integrate them with existing software. The model is also programming language-neutral, so developers can use virtually any language they prefer [Microsoft.com, 1999].

The Windows 2000 product line consists of four products:

- Windows 2000 Professional, aimed at individuals and businesses of all sizes. It includes security and mobile use enhancement. It is the most economical choice.
- Windows 2000 Server, aimed at small-to-medium size businesses. It can function as web server and/or a workgroup (or branch office) server. It can be part of a two-way symmetric multiprocessing system. NT 4.0 servers can be upgraded to this server.
- Windows 2000 Advanced Server, aimed at being a network operating system server and/or application server, including those involving large databases.
- Windows 2000 Datacenter Server, designed for large data warehouses, online transaction processing (OLTP), econometric analysis, and other applications requiring high-speed computation and large databases.

2.7.2 Microsoft Windows XP

Overwhelmingly, Windows XP is about the home user. Satisfying a long-standing goal of bringing the stability and reliability of Windows NT/2000 to consumers, Windows XP must also provide the compatibility of Windows 9x while expanding into digital media-themed experiences. Years after the initial push toward an iterative, activity-based user interface in its aborted "Neptune" project, Microsoft has delivered a user interface in Windows XP that delivers on the promise of integrated experiences. Users of Microsoft Windows XP can experience digital photography, digital video, digital music, the internet, mobile computing, the connected home, real-time communications and a variety of other experiences.

2.8 Summary

Literature review on various aspects has gained many ideas for me to develop this project. Through reading, surveying web sites and reviews on several done thesis proposal, manage to gather enough information with guidance.

This literature review is for analysis and to consider the requirements for the system. From the existing web site been review, I can compare it with my proposed system and from there I can analysis it and build a better system from it. Besides, from this chapter I have do analysis on the development platform and development technology. There are many of choices in choosing the suitable latest technologies and VRML development platform. So, I will choose the most suitable development platform and development technology. I will state more clearly the conclusion on choosing such development in chapter 3.

CHAPTER THREE: SYSTEM METHODOLOGY

Chapter 3: System Methodology

3.1 Introduction

The previous chapters have introduced the overview and information gathered in the literature search and review where as this chapter would further elaborate the justifications for the chosen project methodology. System methodology is very important in order to make sure that the project has been well planned from the beginning stage until the end of this project. To guarantee success of this project, research has been done on the related fields and system planning based on the approaches provided. To achieve this compatibility and to provide a framework for life-cycle automation, a pattern of thought or paradigm must be established. An example of a paradigm is the traditional waterfall model of development.

3.2 Software Development Life Cycles (SDLC)

Software development life cycles are abstract model that define the activities, ordering of activities and information flow associated with the development of software. Life cycles exist for describing the entire development process as well as for particular focused activities such as system test or preparing to release software. These life cycles form the basis of project management. Without a life cycle to depict the activities that software development is supposed to undertake, a project management is not capable the software development has progressed.

3.3 System Development Methodology

Waterfall Model with prototyping has been utilized as a guideline in implementing the project system development. This methodology is very important in order to make sure that the project has been well planned from the beginning stage until the end of the project. To guarantee the success of this project, research has been done on the related fields and system planning based on the approaches provided.

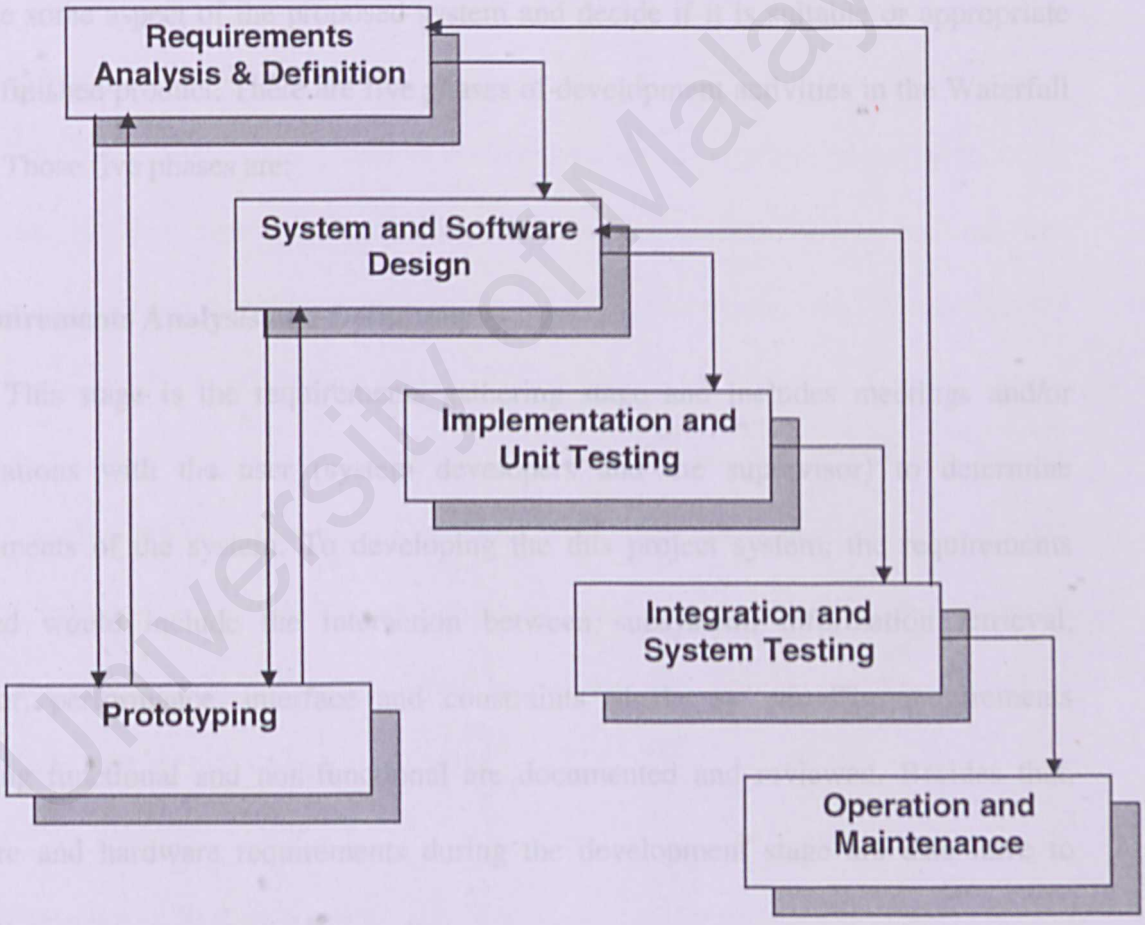


Figure 3.1: Waterfall Model with Prototyping

3.4 Waterfall Model with Prototyping

As the name imply, the stages of development cascades from one phase to another. Each stage of a development is required to be completed before proceeding to the next phase. This model suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing and implementation [Pressman, 2001].

Prototyping is a partially developed product that enables users and developers to examine some aspect of the proposed system and decide if it is suitable or appropriate for the finished product. There are five phases of development activities in the Waterfall model. Those five phases are;

1. Requirements Analysis and Definition

This stage is the requirements gathering stage and includes meetings and/or consultations with the user (system developers and the supervisor) to determine requirements of the system. To developing the this project system, the requirements gathered would include the interaction between subsystem, information retrieval, behavior, performance, interface and constraints of the system. The requirements including functional and non-functional are documented and reviewed. Besides that, software and hardware requirements during the development stage are also have to identify.

2. System and Software Design

The systems design establishes overall system architecture. For this project, system design focuses on distinct attributes of the modules in this project. The system architecture, content design, interface representation, conceptual design and technical of this project are also required in this stage. In addition, the project system design also includes transforming the requirements into a representation that can be assessed for quality before the code generation (implementation stage) begins.

3. Implementation and Unit Testing

During this stage, the software is realized as a set of programs or programs units. Unit testing involves verifying that each unit meets its specification [Ian Sommerville, 2001. Set of programs or program units section in like tranformation will be create in the system based on the project system design. After code has been generated, unit testing is performed to verify that each unit meets its specification. One code has been generated, program testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals that is, conducting tests to uncover errors and ensure that defined input will produce actual results that agree with required results.

4. Integration and System Testing

The individual program units or programs modules are integrated and tested as a complete system to ensure that the software requirements have been met.

5. Operation and Maintenance

Maintenance takes place after the system is put into practical use. Maintenance involves correcting errors missed in earlier stages of the life cycle, improving the system implementation, adding performance or functional enhancements or making changes due to accommodate changes in the software external environment for overall this project.

There are many reasons why the Waterfall Model with Prototyping is the chosen methodology for 3D integration tool development. With Waterfall Model with Prototyping, the strength of each can be achieved on single project. Furthermore, this model can reduce the risk in development by clarifying requirement for the system design before coding. The model is actually the classic Waterfall Model combines with the prototyping approach in its early stages.

3.4.1 Advantages of Waterfall Model with Prototyping [Pfleeger, 1999]

1. This model allow project managers to use the model to gauge how close the project was to completion at a given point in line because associated with each process actually were milestone and deliverable.
2. Its simplicity makes it easy to explain to those who are not familiar with software development.
3. It provided opportunity to explore alternative strategies and revisions.
4. User development in early stage ensures the system is developed more closely to user requirement.

3.5 Summary

In this chapter, study has been carried out on this project to find out the suitable methodology used. Waterfall Model with Prototyping has been chosen for its stability and structure. The next chapter will continue to give a clearer picture of the system by showing the requirements of the system, ranging from functional, non-functional to hardware and software requirements.

CHAPTER FOUR: SYSTEM ANALYSIS

Chapter 4: System Analysis

4.1 Introduction

After the literature search and review, the next step is to perform a detailed analysis. This chapter consists of two parts which are software system requirements and justification on software development tools. The requirements needed for the system development consists of functionality requirements, non-functionality requirements, hardware requirements and software requirements.

Through system analysis, the information gathered during this phase has provided alternative strategies to develop this software. Although software design can be identified and defined as a distinct activity, it must be compatible in both concept and implementation with essential development activities such as analysis and coding that precede or follow it. Through this phase also, the programmer can determine types of functional requirements and non-functional requirements for the software.

4.2 Objectives of System Analysis

Following are some of the objectives of the analysis:

- To identify the major modules to be included in the software.
- To identify what are the modules that are feasible to develop and the knowledge and tools need to have in order to develop them.
- To choose the development tools for the new system among different types of tools that have been studied in Chapter 2.
- To acquire knowledge on how this system will be developed with the new emerging technology.

4.3 System Requirement Analysis

In general, requirement for this project are partitioned into functional requirements and non-functional requirements. Functional requirements are associated with specific functions, tasks or behaviors the system must support, while non-functional requirements are constraints on various attributes of these functions or tasks [Sommerville, 2001].

In terms of the ISO quality characteristics for evaluation (which is concerned primarily with the definition of quality characteristics to be used in the evaluation of software products), the functional requirements address the quality characteristic of functionality while the other quality characteristics are concerned with various kinds of non-functional requirements. Because non-functional requirements tend to be stated in terms of constraints on the results of tasks, which are given as functional requirements (e.g., constraints on the speed or efficiency of a given task), a task-based functional requirements statement is a useful skeleton upon which to construct a complete requirements statement. That is the approach taken in this work.

4.3.1 Functional Requirement

Functional requirement is a statement of the services or functions that a system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. The following are the functional requirements for this system.

- The user able to perform functions such as integration, scaling, and rotation to the 3D objects in a virtual environment.

- The software should allow the user to acquire 3D files from other sources or provided by the software.
- The software able to perform save function to the 3D environment after the 3D object's integration and/or transformations.

4.3.2 Non-functional Requirement

Non-functional requirement are as important as functional requirements. It is defined as constraints under which the system must operate and the standard, which must be met by the delivered system.

The non-functional requirement of the project system specification is presented as below:

a. Reliability

Reliability is the extents to which a system can expect to perform its intended functions with required precision and accuracy. Reliability is also responsible to convince the users that the system developed will make the correct respond and provide error-handling ability. The system should not contain bugs after implementation of the operational system.

b. Scalability

The scalability is to promise the capability of the system to migrate as a client or server to machines of greater or lesser power, depending upon requirements, with little or no change to underlying components.

c. Usability

The system should be developed in such a way that it is easy to use. It will enhance and support rather than limit or restrict the original process. Human interface need to be designed as intuitive and consistent as it could be with other modules in the environment and within themselves.

d. Maintainability

The system should be designed in such a way that it can be easily understood, and corrected, as well as adapted and allow enhancement in the future. It is of no benefit to develop a system that has no potential for improvements.

e. Manageability

The modules within the system should be easy to manage. This will make the maintenance and enhancement works simpler and not times consuming.

f. Simplicity and User Friendly

The system is required to have a very user-friendly interface because most of the users are computer illiterate. The usage of suitable and meaningful captions and icons will help the user to consume the system with more confidence, easy and save time. An attractive and easily understand user interface is needed in a system. It should be a neutral design based on the level of all 3D users.

4.3.3 Software Requirement

In order to host and develop the full system, the developer's computer system must have the supporting software as listed below:

Description	Technologies/Software
Operating system/ Platform	Microsoft Window XP
Programming Languages	Microsoft Visual Basic, C++
Hardware API	OpenGL

Table 4.1 Software Requirements

4.3.4 Hardware Requirement

The recommended hardware configuration is as follows:

Component	Description
Microprocessor	Intel Pentium 355 MHz and above
RAM	64 MB RAM and above
Storage	2 GB hard disk with a minimum of 250MB of free space
3D Graphic Accelerator Card	256-bit graphics accelerator, AGP 4X support.
Others	Keyboard, mouse, monitor,

Table 4.2 Hardware Requirements

4.4 Tools and Technology Proposed

The main task in this section is to choose suitable development technologies and programming languages that will be used to develop the system. After all the requirements have been reviewed and analyzed, the most suitable and appropriate tools for developing the system are identified and selected. The tools to be selected include the development software as well as the entire platform on which the development of the project is going to be done. The major criteria to be considered are not only the suitability of the tools perhaps; the tools to be used must be able to interact with each other. The following programming technology and languages are chosen in order to develop this project.

4.4.1 Operating System and Platform : Microsoft Window XP

Windows XP will be chosen as the platform to develop this software due to familiarity of use, ease-to-use, and its compatibility with other software. Windows XP is more stable than Windows 2000. It is less likely to clash. Besides, there are so many new features included with Windows XP that will be a significant consumer release and will prove to be a better upgrade than Windows 2000 was.

4.4.2 Programming Languages : Microsoft Visual C++ and Microsoft Basic

4.4.2.1 Microsoft Visual C++

Although Java is proven to be the more powerful programming tool here, it is still not as user friendly as C++ especially in command control aspects. Furthermore, Java is known for their difficulty in integration and configuration. As such, it will take considerable time and effort to learn Java, thus making it unsuitable for the development of this project. This is why C++ as programming languages for this project. C++ is recognized as one of the technical programming language in the sense of command and machine operation awareness to a programmer. In other words, a C++ programmer needs to understand more clearly on the structure and flow of the operation in order to write the coding. This tool is mainly used by me as my part of this project to compile OpenGL coding.

4.4.2.2 Microsoft Visual Basic.NET

Microsoft Visual Basic.NET able to develops application using the most readable and easy to write programming language available. Background compilation provides feedback for error detection. It also able to create reusable, enterprise-class code using full object-oriented constructs. Language features include full implementation inheritance, encapsulation, and polymorphism. Structured exception handling provides a global error handler and eliminates spaghetti code. This is the reason Visual Basic will be used in the development of this project.

4.4.3 Hardware API : OpenGL

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications and it has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API). OpenGL standard has language bindings for C, C++, Fortran, Ada, and Java. Features like modeling transformations to lighting and texturing and many other features that makes the chosen hardware API for this project. This tool is mainly used by me to do my part in this project which is more concentrate on 3D environment.

4.5 Summary

This chapter describes the detail of the requirements of the system, ranging from functional, non-functional to hardware and software requirements. Besides that, development tools and technology are also has been elaborated. The next chapter will continue to give a clearer picture of the system by showing the system structure and design, data flow diagram, together with the interface and database designs.

CHAPTER FIVE: SYSTEM DESIGN

Chapter 5: System Design

5.1 Introduction

System design is a creative process of transforming the problem into solution and the description of the solution. The goal of system design is to translate the requirements defined during the system analysis phase into a working model or representation of an entity that will be built later. During this phase, quality is fostered. System design involves designing of program and user interfaces. System design has to go through a thorough modification and testing before coming to a complete system. Amendment has to be done on every occurrence of mistakes especially in coding and user interfaces design. Under this chapter, the system design will be discussed into the following few components:

- System Functionality Design
- Graphical User Interface Design
- Scene Graph

5.2 System Functionality Design

The system architecture, data flow diagram, and system structure chart for this project will be discussed under the system functionality design section.

5.2.1 System Architecture

This project system architecture is based on the stand alone system architecture. This common system architecture is divided into three layers. There are graphical user interface (GUI) layer, programming language layer and hardware API layer. The first layer related to GUI will be developed by my partner who concentrates more on drag n drop icons on the icon panel and menu control panel. She shall most probably using Visual Basic.net to develop such interfaces.

The second layer is concerning programming language as a main role to this project to integrate both first and third layer. I shall develop this particular layer and concerns about the relation between the first and second layer.

The third layer will be developed as a group work for both of us which concentrates the API. These interfaces provide access to the support functions defined in the classes as well as provide additional useful functionality. The API also does the actual sending and receiving of raw data for the application, but leaves it up to the networking software and middleware to address the packet and take care of computer semantics.

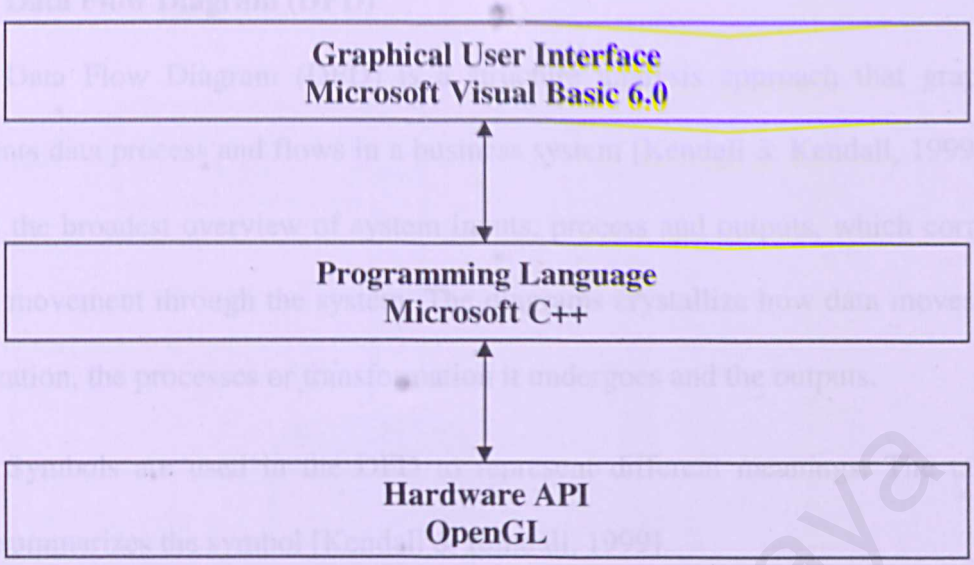


Figure 5.1: Three Layers in System Architecture

5.2.2 Data Flow Diagram (DFD)

Data Flow Diagram (DFD) is a structure analysis approach that graphically represents data process and flows in a business system [Kendall & Kendall, 1999]. DFD depicts the broadest overview of system inputs, process and outputs, which correspond to data movement through the system. The diagrams crystallize how data moves within organization, the processes or transformation it undergoes and the outputs.

Symbols are used in the DFD to represent different meanings. The table 5.1 below summarizes the symbol [Kendall & Kendall, 1999].

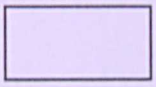
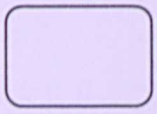
Symbols	Meaning	Explanation
	Entity	A person, group, department or other system that can send data or receive data from the system
	Process	A transformation of data
	Data Store	A repository for data that allows addition and retrieval of data
	Data Flow	Movement of data from or to one process

Table 5.1: Symbols in Data Flow Diagram (DFD)

5.2.2.1 Context Diagram

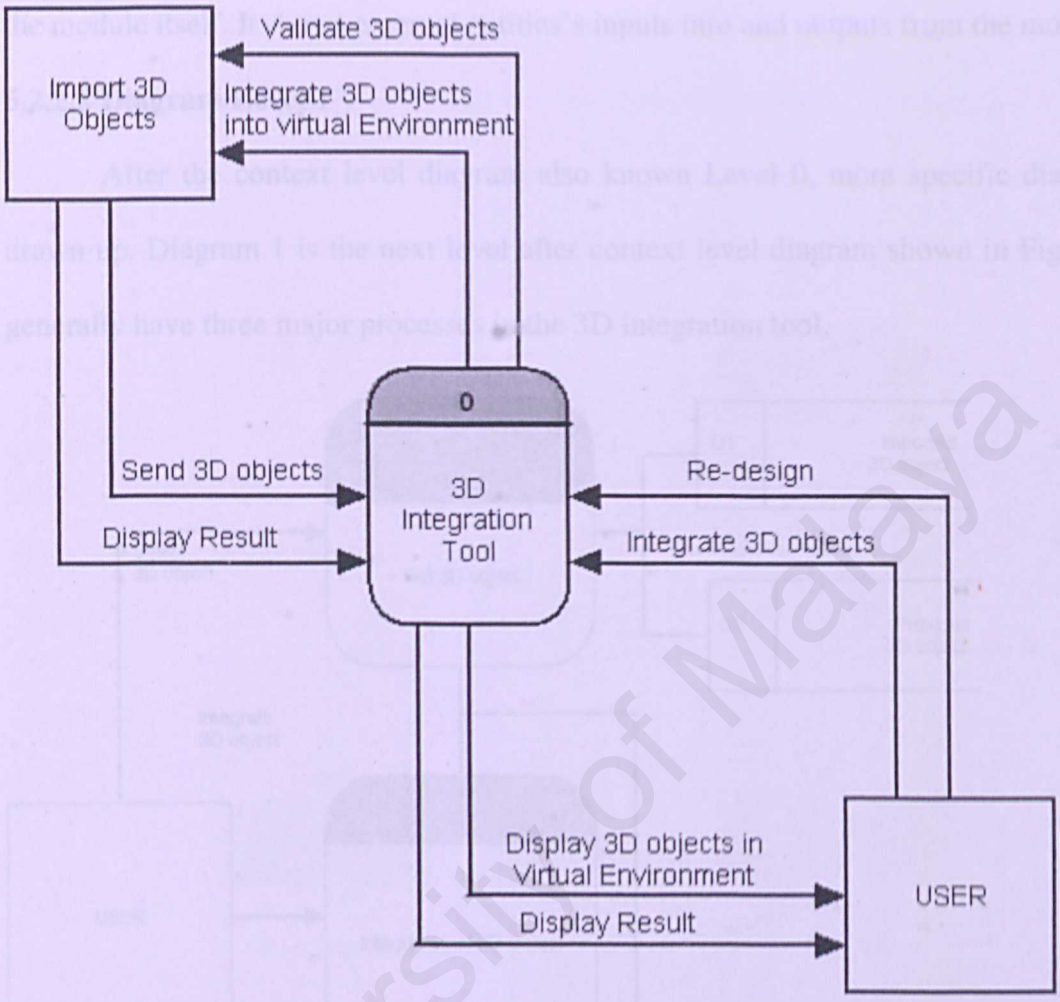


Figure 5.2: Context Diagram for 3D integration tool.

A context data flow diagram defines the scope and boundary for the system and project. Because the scope of any project is always subject to change, the context diagram also subject to constant change.

The context diagram in Figure 5.2 is the most general diagram, a bird-eye's view of data movement in the module and the broadest possible conceptualization of the system. It shows the external entities inputs, the general module and possible outputs

from the module. It is the highest level in DFD and contains only one process, which is the module itself. It shows external entities's inputs into and outputs from the module.

5.2.2.2 Diagram Level 1

After the context level diagram also known Level 0, more specific diagram is drawn up. Diagram 1 is the next level after context level diagram shown in Figure 5.3; generally have three major processes in the 3D integration tool.

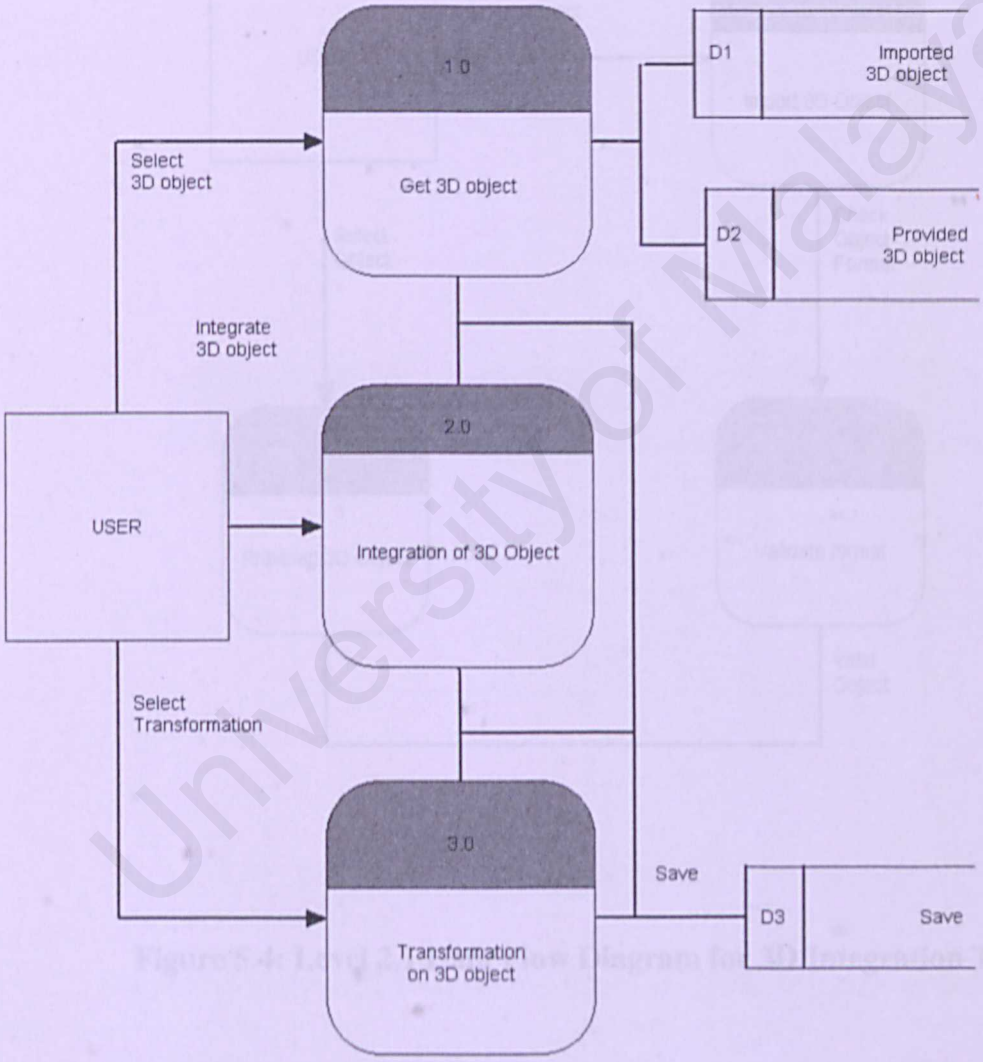


Figure 5.3: Level 1 Data Flow Diagram for 3D Integration Tool

5.2.2.3 Child Diagrams

Each of child diagrams is from the major processes in the Diagram 0 of the 3D integration tool module. As what can be seen from the diagram, Process1.0, 2.0 and 3.0 Diagram 0 are then expanded to create a more detailed child diagram. The child diagrams represent new lower level data flow. It is illustrated in Figure 5.4, and 5.5.

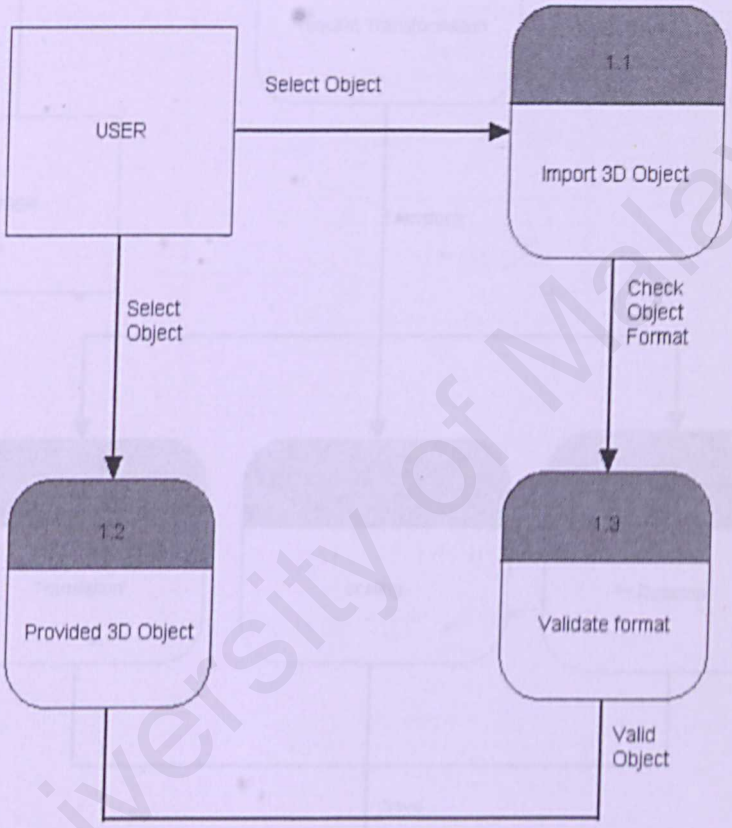


Figure 5.4: Level 2.1 Data Flow Diagram for 3D Integration Tool

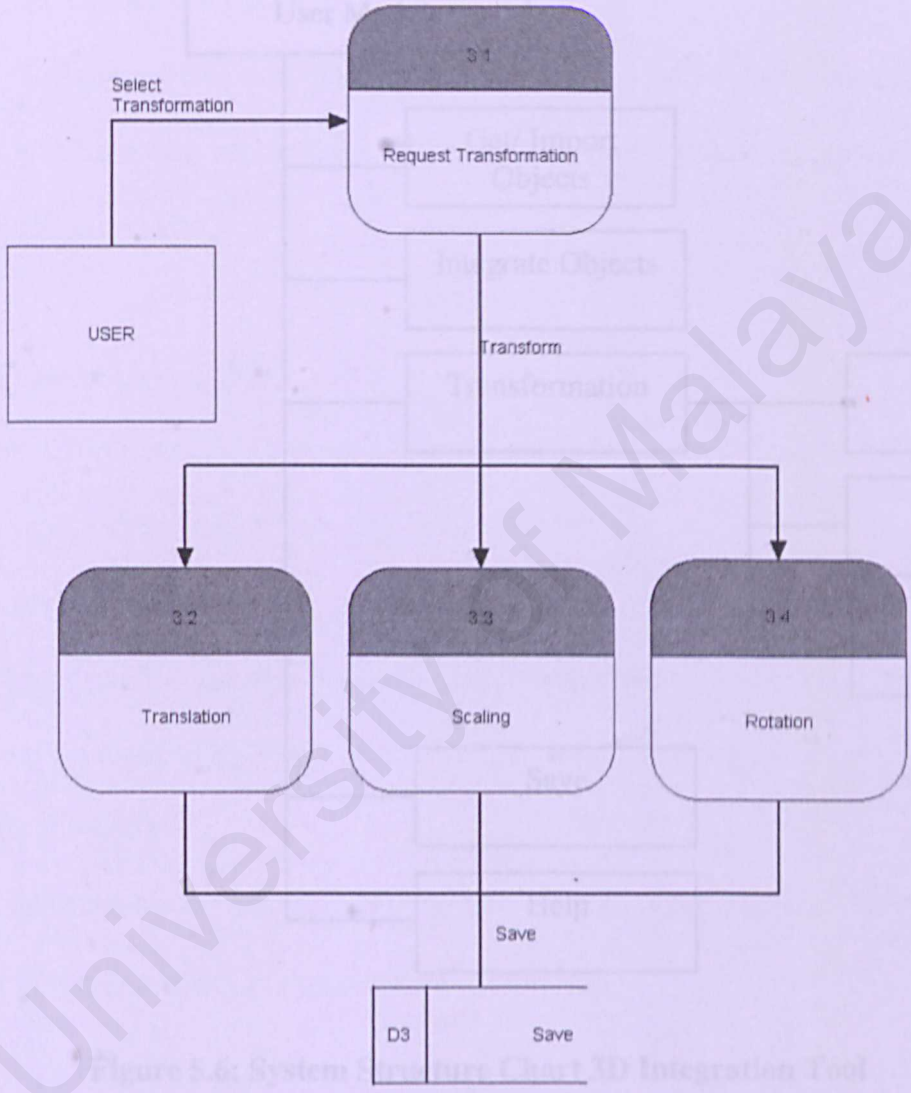


Figure 5.5: Level 2.2 Data Flow Diagram for 3D Integration Tool

5.2.3 System Structure Chart

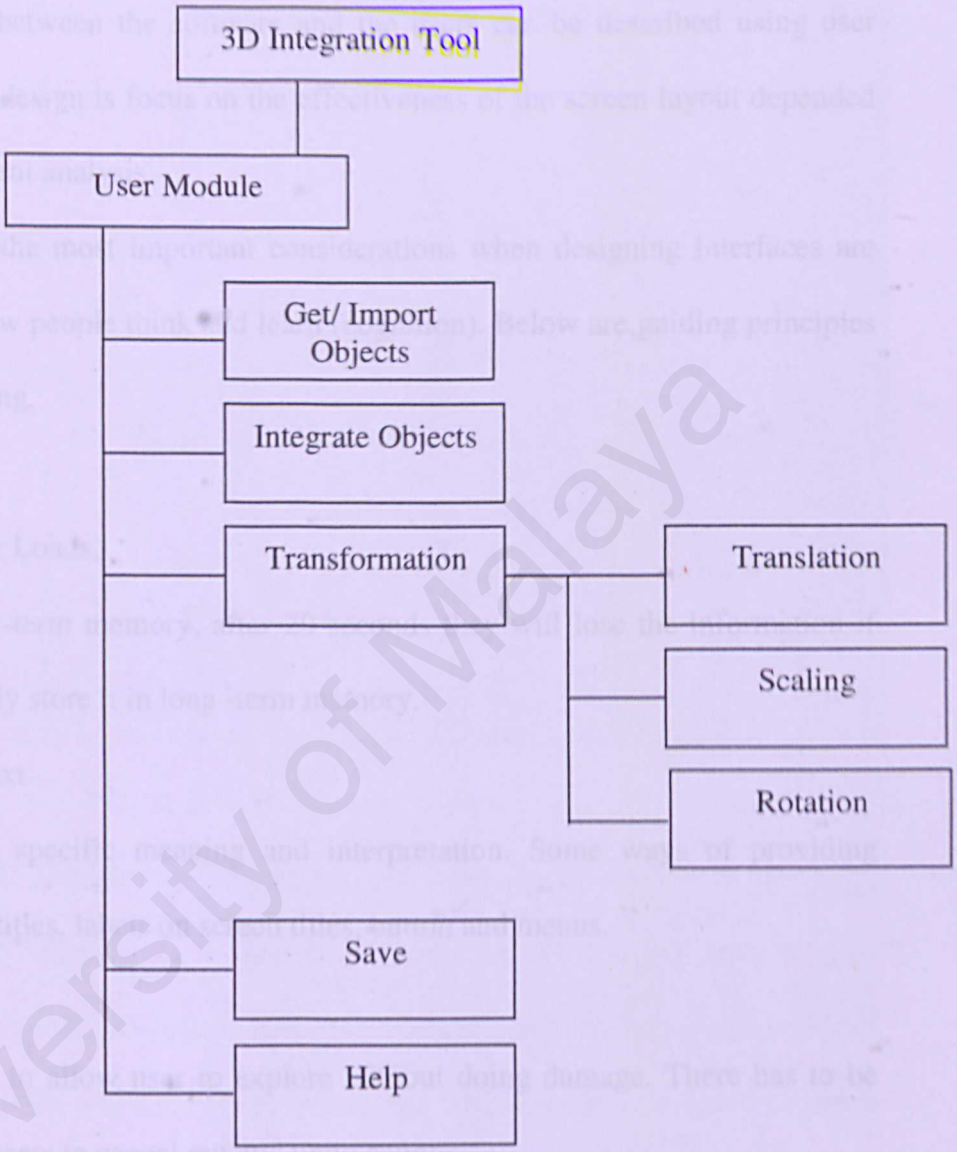


Figure 5.6: System Structure Chart 3D Integration Tool

This system structure chart performs different system functions in 3D integration tool. The module provides much functionality to the users. The users can get or import 3D files and integrate them with each others. After users decide to do transformation, they also can save the files or use some help function.

5.3 Graphical User Interface Design

Relation between the software and the users can be described using user interface design. The design is focus on the effectiveness of the screen layout depended on the users requirement analysis.

Some of the most important considerations when designing interfaces are the ones involving how people think and learn (cognition). Below are guiding principles for cognitive processing.

a) Limit Memory Loads

People have short-term memory, after 20 seconds they will lose the information if they cannot quickly store it in long -term memory.

b) Provide Context

Context provides specific meaning and interpretation. Some ways of providing context are using titles, labels on screen titles, button and menus.

c) Be forgiving

The interface has to allow user to explore without doing damage. There has to be built in ways for users to cancel out and undo actions.

5.3.1 Design of Screen

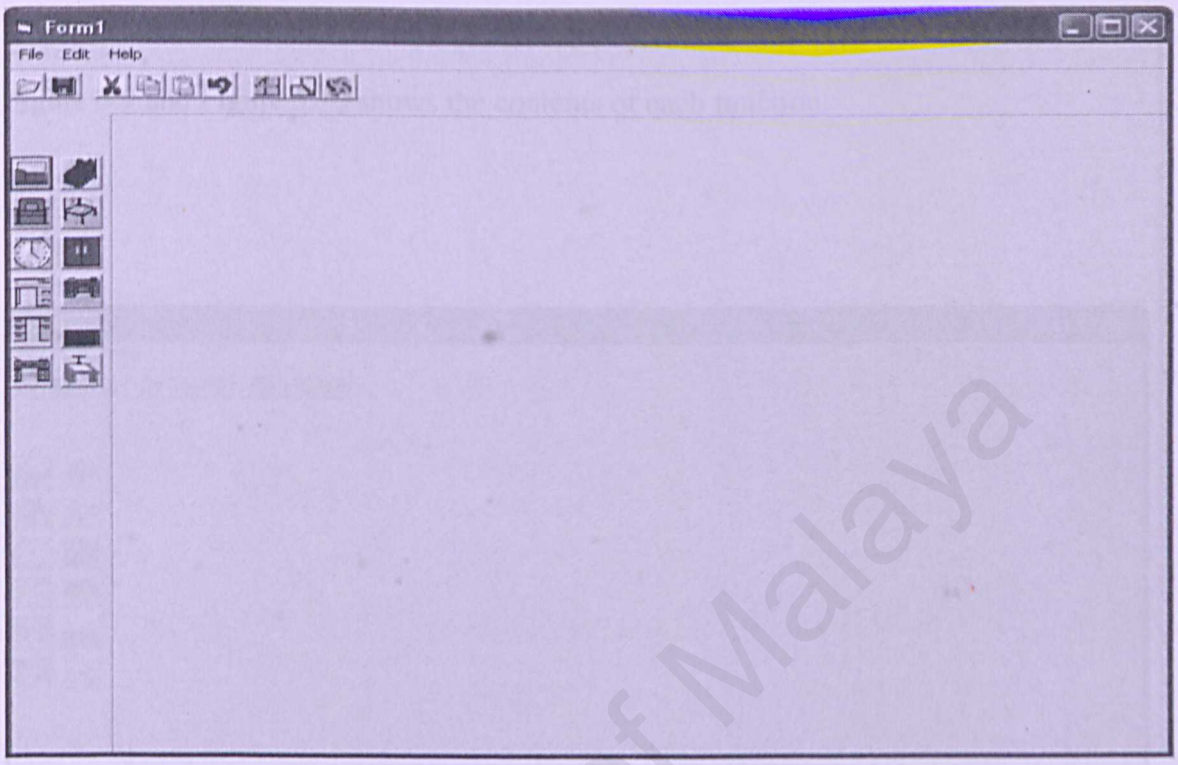


Figure 5.7: The Screen Design

There are Left Navigation, Top Menu and Main Body section in the main screens. The Left Navigation is a section consists of menus for features like transformation and provided 3D objects such as table and bed. The Top Menu is where the main function like file, edit and help function. The virtual environment will be in the Main Body.

5.3.2 The Top Menu

The Top Menu consists function like file, edit and help function. Figure 5.8, Figure 5.9 and Figure 5.10 shows the contents of each function.

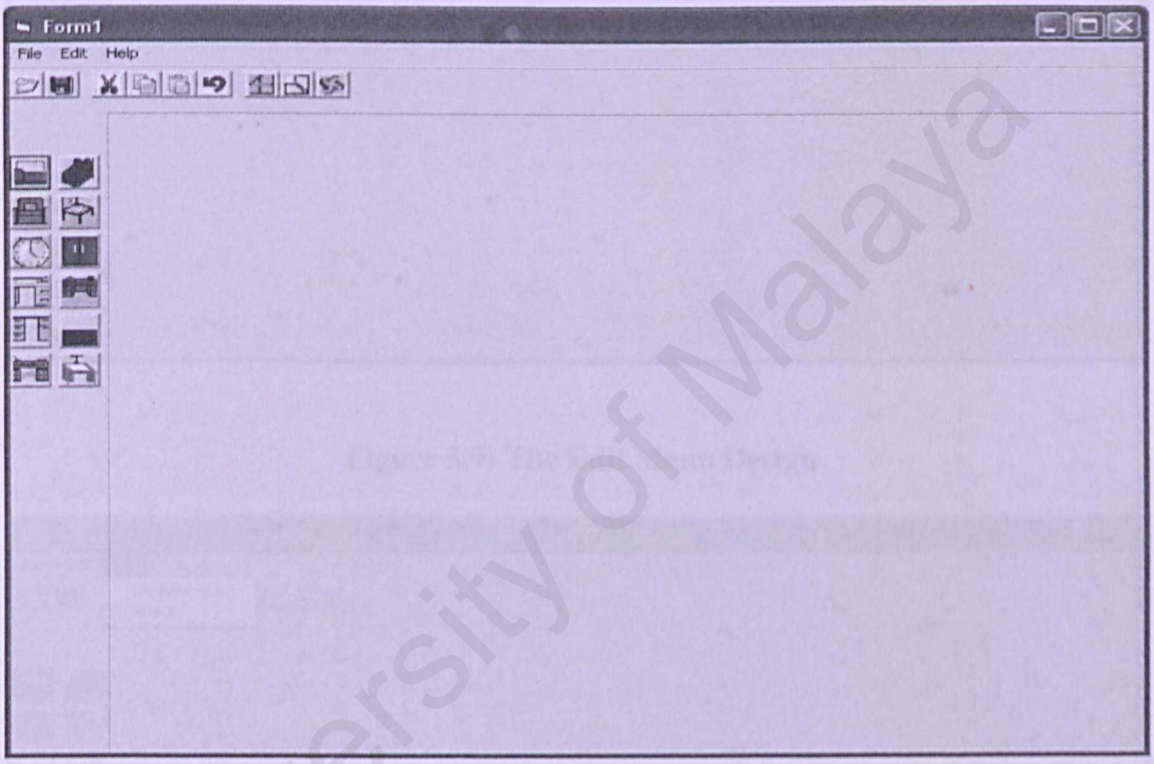


Figure 5.8: The File Menu Design

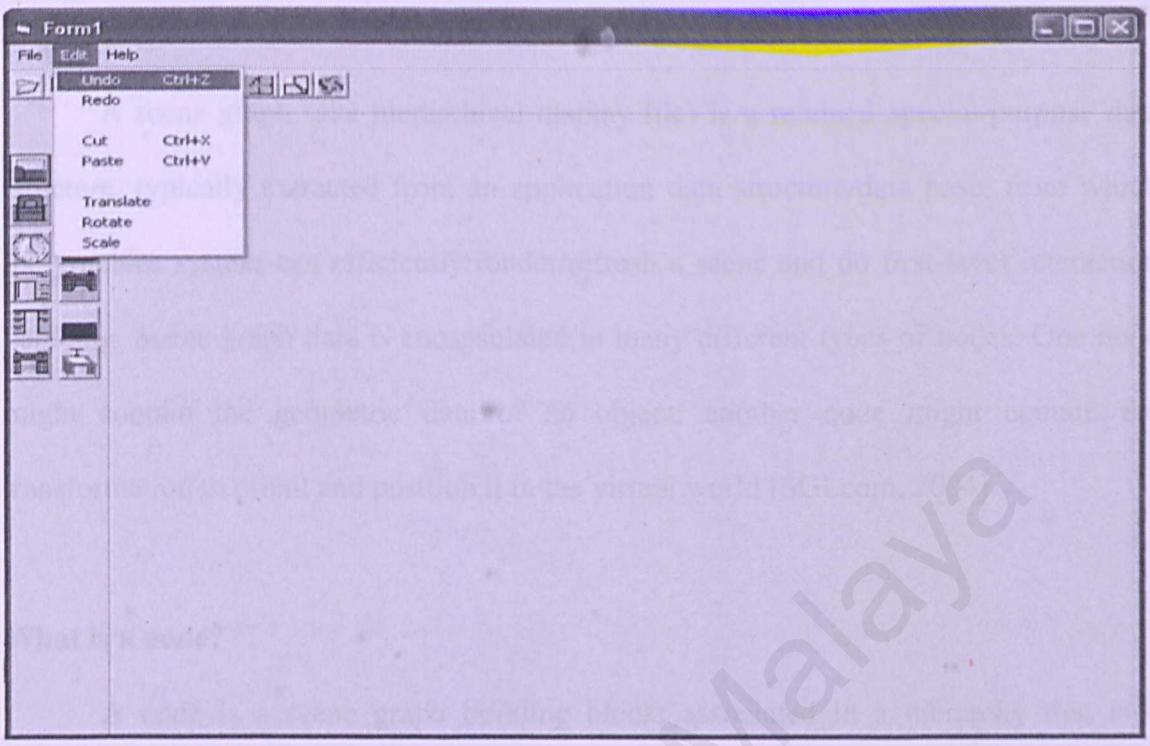


Figure 5.9: The Edit Menu Design

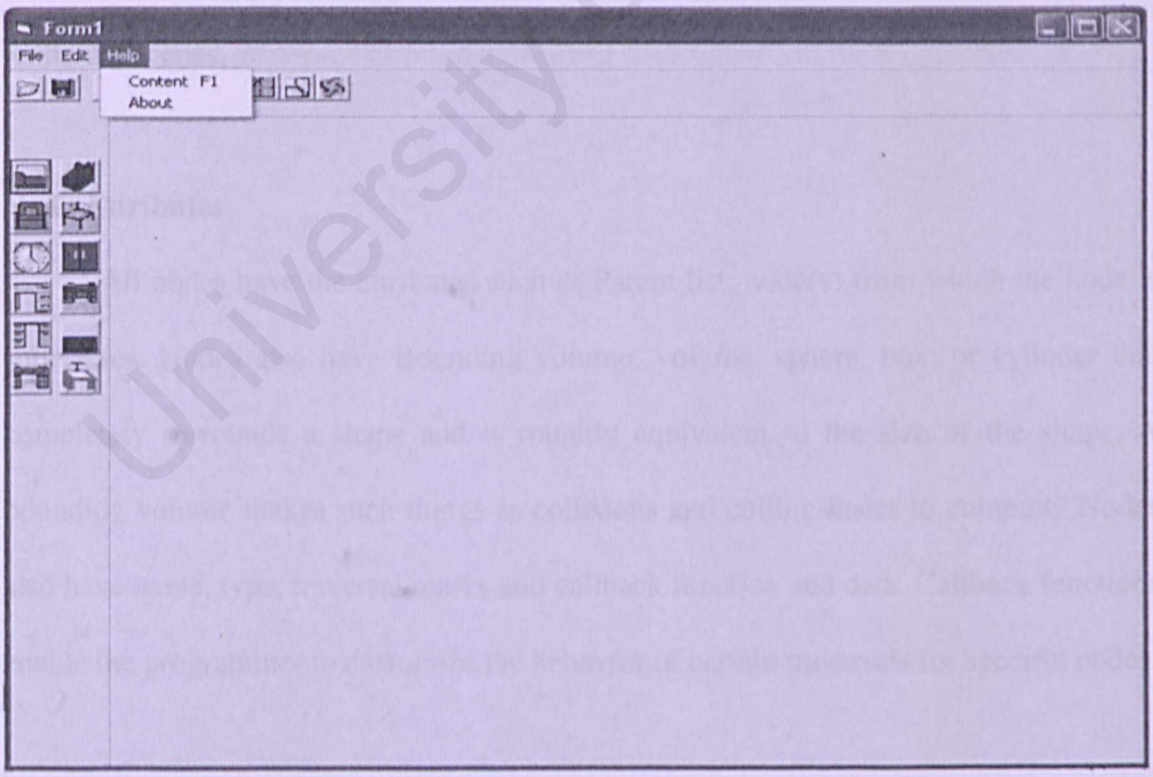


Figure 5.10: The Help Menu Design

5.4 Scene Graph

A scene graph (aka hierarchical display file) is a retained special-purpose data structure, typically extracted from an application data structure/data base, from which the graphics system can efficiently render/refresh a scene and do first-level interaction handling. Scene graph data is encapsulated in many different types of nodes. One node might contain the geometric data of an object; another node might contain the transformation to orient and position it in the virtual world [SGI.com, 2004].

What is a node?

A node is a scene graph building block; associated in a hierarchy that is a directed, acyclic graph. OpenGL and other application can act on the scene graph to perform various complex operations efficiently, such as database intersection and rendering scenes.

Node Attributes

All nodes have the attributes such as Parent list; node(s) from which the node is subclasses. Nodes also have Bounding volume; volume, sphere, box, or cylinder that completely surrounds a shape and is roughly equivalent to the size of the shape. A bounding volume makes such things as collisions and culling faster to compute. Nodes also have name, type, traversal marks and callback function and data. Callback functions enable the programmer to customize the behavior of certain traversals for specific nodes.

What is scene graph node?

The two most general classifications of node functionality are:

- Group nodes

Associate nodes into hierarchies, only group nodes can have child nodes. Each child node has an index number; the first child added to a group node has an index number of 0, the next child added has an index number of 1, and so on. The group node keeps a list of its child nodes. A child node may have multiple parent nodes.

- Leaf nodes

Leaf nodes contain all the descriptive data of objects in the virtual world used to render them. Leaf nodes also contain the descriptive values used to render all the visual elements in the virtual world. Leaf nodes cannot have child nodes. For example, a node encapsulating a table lamp might have three parent nodes, each show the type of transformation has to be done to the table lamp before placing it on a table, as shown in Figure 5.11.

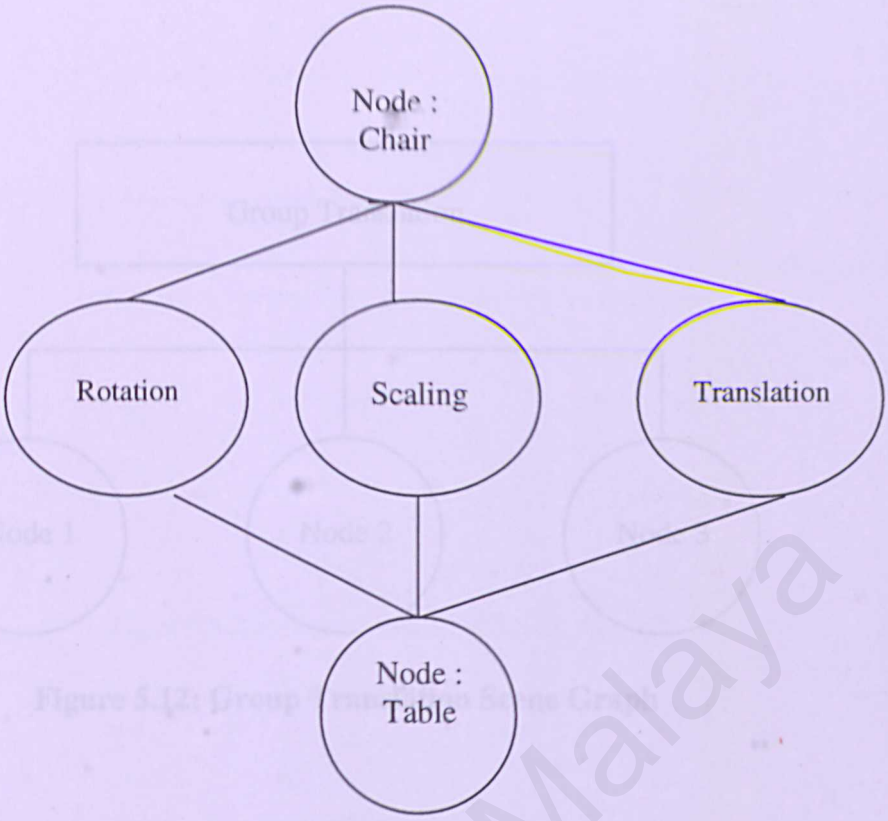


Figure 5.11: Multiple Parent Nodes

In this project, the nodes will be the 3D objects. If the user wants to transform more than one object at one time, they need to put all the objects in one group first. After all the objects has been put in one group, all the objects will be under one command as shown in Figure 5.13, Figure 5.14 and Figure 5.15. Figure 5.16 shows the whole basic flow of nodes in this project scene graph.

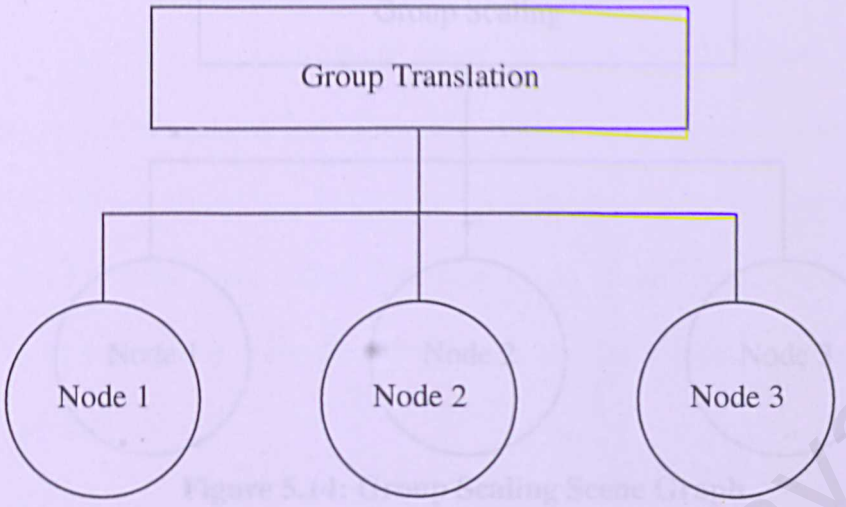


Figure 5.12: Group Translation Scene Graph

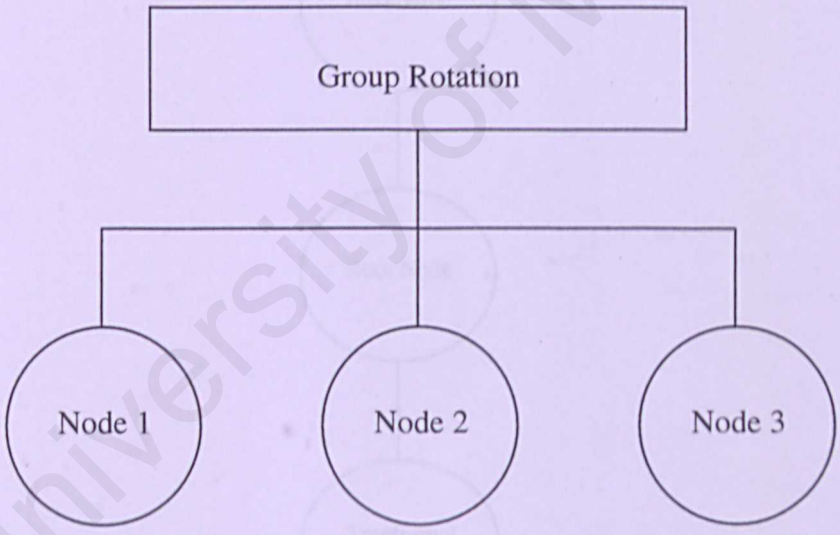


Figure 5.13: Group Rotation Scene Graph

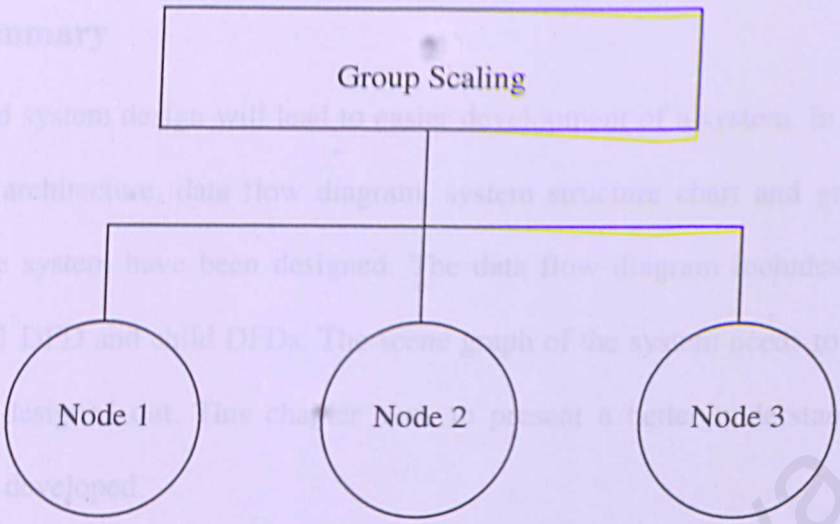


Figure 5.14: Group Scaling Scene Graph

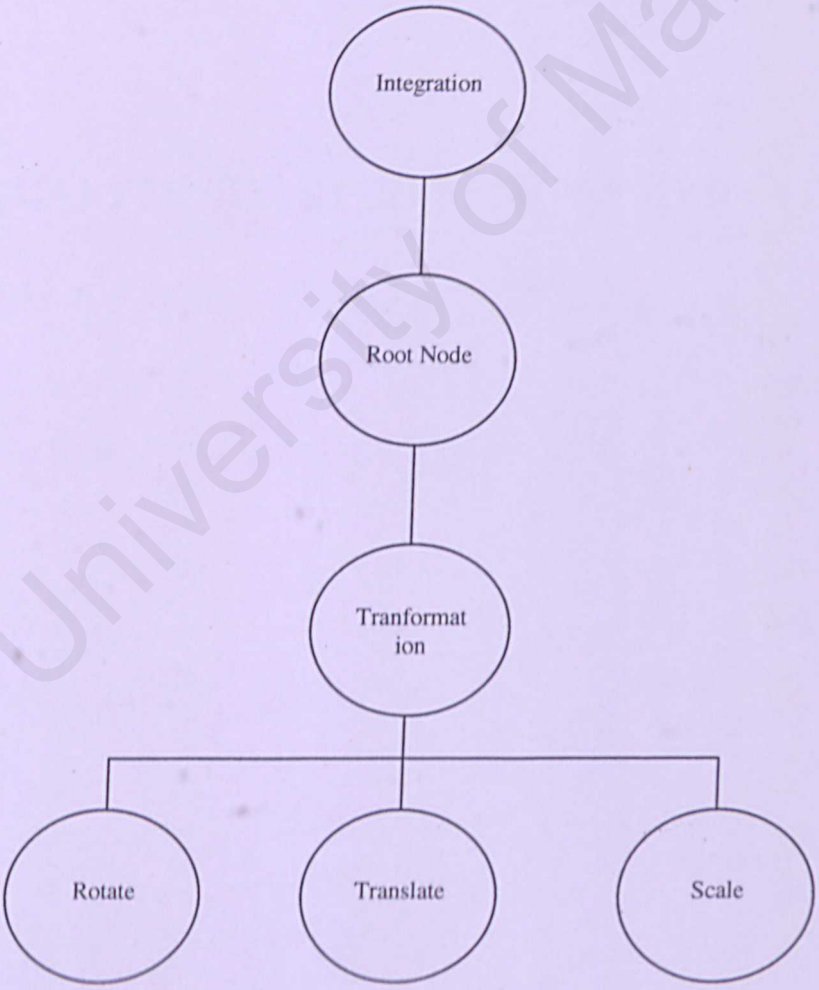


Figure 5.15: Scene Graph Design for 3D Integration Tool

5.5 Summary

A good system design will lead to easier development of a system. In this chapter, the system architecture, data flow diagram, system structure chart and graphical user interface the system have been designed. The data flow diagram includes the level 0 DFD, level 1 DFD and child DFDs. The scene graph of the system needs to be resolved is carefully designed out. This chapter aims to present a better understanding of the system to be developed.

CHAPTER SIX: SYSTEM IMPLEMENTATION

Chapter 6: System Implementation

6.1 Introduction

This chapter will explain the development of the system or model, which is based on the transition of modules and algorithms derived from the system design phase, into feasible instructions by coding it into software. During this phase, the design model of 3D Integration tool is transformed into a workable product. This phase involves coding of the program by using the appropriate programming language and coding approach, testing of the system to ensure every function works properly and debugging the code, which will identify and correct bugs within the program.

Before the coding starts, communication should be set up and several installations, configurations and settings need to be done, especially for providing the features in the 3D Integration tool. When applicable, each installation that is carried out will be discussed relating to the features being used in the 3D Integration tool. The set up is documented in the user manual.

6.2 Development Environment

Development environment has certain impact on the development of a system. Using the suitable hardware and software will not only help to speed up the system development but also determine the success of the project. After implementing the system, the requirement of hardware and software that was stated in the previous chapter (Chapter 4) can be finalized. The final list of the hardware and software tools used to develop the entire system is listed below.

6.2.1 Actual Hardware Requirements

The hardware used to develop the system is as listed below:

- 200MHz Pentium Processor
- 128MB RAM
- 52x CD-ROM Drive
- 10.4 GB Hard Disk Drive
- Other standard desktop PC components

6.2.2 Actual Software Tools Requirements

6.2.2.1 Software Tools for Design and Report Writing

There are a lot of software tools, which can be used in designing and writing report. The design process involves the drawing of structure chart, data flow diagram and others that form the foundation of the software development. The purpose of this graphically logical design is to provide an overall view of system and interconnection between the modules. Visio Professional and Microsoft Word are the software that used to design and write report.

6.2.2.2 Software Tools for Development

Tools/software used for development includes those meant for:

i. Operating System : Ms Windows XP Professional (SP2)
ii. Tools for Coding – Ms Visual C++ 6.0

6.3 Program Development and Coding

Program development is the process of creating the programs needed to satisfy software requirements. Developing and Coding is the phase which takes the longest time in the development life cycle. Therefore, using the right tool and the right way to develop the system are crucial in determining the success of a project. For this project, it involves developing using programming languages from OpenGL, GLUI and C++. Before starting on the coding process or any other detailed works on the program, a review on the program documentation needs to be done followed by design of the program and finally going into the program coding process.

6.3.1 Review the Program Documentation

The first and foremost step to be taken in program development phase is to review the program documentation that was prepared during the earlier phases. The program documentation prepared in the system design phase of this project consists of architectural view, concepts and controls, module flow diagram and also the sample layout of the interface. The documentation provides a guide and an understanding of the works that need to be done in the coding phase.

6.3.2 Designing the Program

After reviewing the program documentation, designing the program is the next following process after that. For this phase, determining how the program can accomplish the features and functions that are described in the program documentation and developing a logical solution to the programming problem is done. The logical solution or the logic of the program is a step-by-step solution to the programming problems.

6.3.3 Coding Approach

Coding is an iterative process whereby it is done until the programmer obtains the desired results. There are two approaches in coding, namely top-down and bottom-up. The bottom-up coding is based on coding some complete lower level modules and leaving the high-level modules merely as skeletons that are used to call the lower modules, whereas the top-down approach is the reverse.

This 3D Integration tool was developed modularly using both the top-down and bottom-up approaches. By combining both approaches at different stages of coding,

testing could be done on those completed modules while others are still being coded. In addition, critical functions can be coded first to test their efficiency. Example of the coding can be found in Appendix. Comments are added in the code to provide a better understanding to the code.

6.3.4 Coding Style

Coding style is an important attribute of source code. An easy to read source code makes the system easier to maintain and enhance. Elements taken into considerations while coding an easy to maintain and enhance system are internal documentation, standard naming convention and standard graphical user interface.

Internal documentation is achieved by using comments while coding, providing a clear guide to programmers for future enhancement. Statements of purpose indicating the functions of modules and descriptive comment are embedded into source code to describe the processing functions.

A standard naming convention and also a standard usage of graphical user interface components is employed in developing the system making. Standard naming convention provides programmers with easy identification of variables. While a standard in usage of graphical user interface components provides the users an environment that will not generate much surprise to them. Usages of these standards perform as a mean towards coding consistency and standardisation.

6.3.5 Chosen OpenGL

Before getting into the intricacies of using OpenGL, we will begin by making a few comments about the philosophy behind the OpenGL API and some of the caveats that come with it.

OpenGL is a procedural rather than descriptive interface. In order to get a rendering of a red sphere the program must specify the appropriate sequence of commands to set up the camera view and modelling transformations, draw the geometry for a sphere with a red color and so on. Other APIs such as Renderman are descriptive and one would simply specify that a red sphere should be drawn at coordinates (x,y). The disadvantage of using a procedural interface is that the application must specify all of the operations in exacting detail and in the correct sequence to get the desired result. The advantage of this approach is that it allows great flexibility in the process of generating the image. The application is free to make tradeoffs between rendering speed and image quality by changing the steps through which the image is drawn. The easiest way to demonstrate the power of the procedural interface is to note that a descriptive interface can be built on top of a procedural interface, but not vice-versa.

A second aspect of OpenGL is that the specification is not pixel exact. This means that two different OpenGL implementations are very unlikely to render the same images. The motivation for this is to allow OpenGL to be implemented across a range of hardware platforms. If the specifications were too exact, it would limit the kind of hardware acceleration that could be used and limit its usefulness as a standard. In practice, the lack

of exactness need not be a burden - unless you plan to build a rendering farm from a set of machines with different implementations.

The lack of pixel exactness shows up even within a single implementation, in that different paths through the implementation may not generate the same set of fragments, although the specification does mandate a set of invariance rules to guarantee repeatable behavior across a variety of circumstances. A concrete example that one might encounter is an implementation that does not accelerate texture mapping operations, but otherwise accelerates all other operations. When texture mapping is enabled, the fragment generation is performed on the host. As a consequence all other steps that precede texturing are also likely to occur on the host, possibly resulting in either different algorithms being invoked or the use of arithmetic with different precision than that used in the hardware accelerator. As a result, when texturing is enabled slightly different pixels in the window may be written as when compared to texturing when it is disabled. For some of the algorithms presented in this course such variability can cause problems, so it is important to understand a little about the underlying details of the OpenGL implementation you are using.

6.4 Implementation

This software has several main parts. My parts are more concern on making and display the 3D objects and 3D environment and also provide transformation function. Almost all coding are using OpenGL which is contains GLUI and GLUT approach.

6.4.1 GLUI

GLUI is a GLUT-based C++ user interface library which provides controls such as buttons, checkboxes, radio buttons, spinners, and list boxes to OpenGL applications. It is window-system independent, relying on GLUT to handle all system-dependent issues, such as window and mouse management.

The OpenGL Utility Toolkit (GLUT) is a popular user interface library for OpenGL applications. It provides a simple interface for handling windows, a mouse, keyboard, and other input devices. It has facilities for nested pop-up menus, and includes utility functions for bitmap and stroke fonts, as well as for drawing primitive graphics objects like spheres, and teapots. Its greatest attraction is its *window system independence*, which (coupled with OpenGL's own window system independence) provides a very attractive environment for developing cross-platform graphics applications.

The GLUI User Interface Library addresses this problem by providing standard user interface elements such as buttons and checkboxes. The GLUI library is written entirely over GLUT, and contains *no system-dependent code*. A GLUI program will therefore behave the same on SGIs, Windows machines, Macs, or any other system to

which GLUT has been ported. Furthermore, GLUI has been designed for programming simplicity, allowing user interface elements to be added with one line of code each.

6.4.2 3D Objects

All 3D objects are being display by calling their own specific function like `glutSolidSphere` which is predefined in the glut library. Commands such as **glMatrixMode**(), **glMultMatrix**(), **glRotate**(), **glTranslate**(), and **glScale**() to compose the desired transformations, or you can directly specify matrices with **glLoadMatrix**() and **glLoadIdentity**(). Commands like **glPushMatrix**() and **glPopMatrix**() used to save and restore model view and projection matrices on their respective stacks. Below is an example taken from the project which displays Sphere.

```
glPushMatrix();
glTranslatef( -0.15, 0.6, 0.0 );
glMultMatrixf( sphere_rotate );
if ( wireframe && show_sphere)
  glutWireSphere( .4, segments, segments );
else if ( show_sphere )
  glutSolidSphere( .4, segments, segments );
if ( show_axes )
  draw_axes(.52f);
glPopMatrix();
```

6.4.3 3D environment

Viewing and modeling transformations are inextricably related in OpenGL and are in fact combined into a single modelview matrix. One of the toughest problems newcomers to computer graphics face is understanding the effects of combined three-dimensional transformations. There are alternative ways to think about transformations for example, do you want to move the camera in one direction, or move the object in the opposite direction? Each way of thinking about transformations has advantages and

disadvantages, but in some cases one way more naturally matches the effect of the intended transformation. Calling **glMatrixMode()** with GL_MODELVIEW as its argument prior to performing modeling or viewing transformations.

All viewing and modeling transformations are represented as 4 × 4 matrices. Each successive **glMultMatrix***() or transformation command. The three OpenGL routines for modeling transformations are **glTranslate***(), **glRotate***(), and **glScale***(). These routines transform an object by moving, rotating, stretching, shrinking, or reflecting it. All three commands are equivalent to producing an appropriate translation, rotation, or scaling matrix, and then calling **glMultMatrix***() with that matrix as the argument. However, these three routines might be faster than using **glMultMatrix***(). OpenGL automatically computes the matrices.

Below is an example taken from the project which is the definition of the display function.

```
void myGlutDisplay( void )
{

  glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

  glMatrixMode( GL_PROJECTION );
  glLoadIdentity();
  glFrustum( -xy_aspect*.04, xy_aspect*.04, -.04, .04, .1, 15.0 );

  glMatrixMode( GL_MODELVIEW );

  glLoadIdentity();
  glMultMatrixf( lights_rotation );
  glLightfv(GL_LIGHT0, GL_POSITION, light0_position);

  glLoadIdentity();
  glTranslatef( -0.5, -0.2, -3.6f );
  glTranslatef( obj_pos[0], obj_pos[1], -obj_pos[2]);
  glMultMatrixf( view_rotate );

  glScalef( scale, scale, scale );
...
```

6.4.4 Transformation Function

6.4.4.1 Rotation Controls

Rotation controls allow the user to input rotations into an application by manipulating an arc ball control. The control displays as a checkerboard-textured sphere, which the user manipulates directly. The current rotation of the control is passed to the application as an array of 16 floats, representing a 4´4 rotation matrix. This array can be passed to OpenGL directly to rotate an object, using the function **glMultMatrix**(). Note that because this control deals with pure rotations only (no translation or scaling), the transpose of the 4´4 matrix is the same as its inverse.

Alternatively, the control and its associated live array can be initialized to the identity matrix by calling **GLUI_Rotation::reset**() after the rotation control is created. The rotation control can be constrained to horizontal-only movement by holding the CTRL key or to vertical rotation by holding the ALT key.

Rotation controls can optionally keep spinning once the user releases the mouse button. To enable this, use the function **GLUI_Rotation::set_spin**(). Note that spinning should be disabled in performance-critical applications (it is disabled by default).

Rotation control

Below is an example taken from the project which is part of the rotation controls.

```
GLUI_Rotation *view_rot = glui2->add_rotation( "Objects", view_rotate
);
  view_rot->set_spin( 1.0 );

//"Objects" will rotate all the 3D objects simultaneously
```

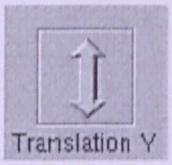
```
  GLUI_Rotation *sph_rot = glui2->add_rotation( "Sphere", sphere_rotate
);
  sph_rot->set_spin( .98 );

//This will only rotate the "Sphere" object
```

6.4.4.2 Translation Controls

Translation controls allow the user to manipulate X, Y, and Z values for 3D objects by clicking on on-screen arrows. The rate of change of translation can be varied by holding down SHIFT for fast movement (100 times faster), or CTRL for slow movement (100 times slower). The function **GLUI_Translation::set_scaling**() can be used to set an overall movement scaling factor.

The four types of translation controls are shown below. When using the XY control (on the left), movement can be restricted to a single axis (either X or Y) by holding down ALT and clicking on either the horizontal or the vertical arrows.



Below is an example taken from the project which is part of the translation controls.

```
GLUI_Translation *trans_xy =
 glui2->add_translation( "Objects XY", GLUI_TRANSLATION_XY, obj_pos);
trans_xy->set_speed( .005 );
glui2->add_column( false );

GLUI_Translation *trans_x =
 glui2->add_translation( "Objects X", GLUI_TRANSLATION_X, obj_pos );
trans_x->set_speed( .005 );
```

6.5 Summary

In this system implementation phase, nearly all the design phases that have been presented and directed toward a final objective that needs to translate representation of system into a form that can be understood by computer. Overall, the primary goal of this phase is to produce a simple, clear source code with internal documentation that will ease the processes of a verification, debugging, testing, modification and further enhancement.

CHAPTER SEVEN: SYSTEM TESTING

Chapter 7: System Testing

7.1 Introduction

After the development and coding in implementation phases, this is followed by the system testing stage. Testing is done throughout the system development and not just at the end. All the system's newly written or modified application program as well as procedural manuals, hardware and system interfaces are tested thoroughly. Testing also meant to turn up heretofore unknown problems. Testing is an essential series of steps that helps assure quality of the system. It is done on many different levels at various intervals as work progresses.

Many programmers view testing as a way to demonstrate how their program performs properly. However, the idea of demonstrating correctness is really the reverse of that testing is all about. We test a program to demonstrate the existence of a fault. Because our objective is to find faults, we consider a test successful only when a fault is discovered. This is achieved by using carefully planned test strategies and realistic data so that the entire testing process can be methodically and rigorously carried out.

System testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and code generation. In this chapter, system testing fundamentals, testing strategies and software debugging methods will be presented.

Following are some of the objectives of system testing:

- Testing is a process if executing a program with the intent of finding an error.
- A good test case is noted that has a high probability of finding an as-yet-undiscovered error.
- A successful test is one that uncovers as-yet undiscovered error.

7.2 Types of Fault

The objective of testing is to find error and fault. Fault identification is the process of determining what fault or faults caused the failure, and fault correction or removal is the process of making changes to the system so that the fault are removed.

When no obvious fault exists, program is tested to isolate more faults by creating conditions where the code does not react as planned. Therefore, it is important to know kind of faults to seek.

Faults can be categorized as below:

1. Algorithmic faults
2. Syntax faults
3. Documentation faults

7.2.1 Algorithmic Fault

Algorithmic faults occur when a component's algorithm or logic does not produce the proper output for given input because something is wrong with the processing steps. These faults are easy to spot by reading through the program (call desk checking) or by submitting input data from each of the different classes of data that we expect the program to receive during its regular working.

Typical algorithmic faults include:

1. Testing for the wrong condition.
2. Forgetting to initialize variables or set loop invariants.
3. Forgetting to test for a particular condition (such as when division by zero might occur).

7.2.2 Syntax Fault

Syntax faults can be checked while parsing for algorithmic faults. This will ensure that the construct of programming language is used properly. Microsoft Visual C++ does come with a compiler to catch syntax faults. Therefore, syntax faults can only be traced after the compiling process finished.

7.2.3 Documentation Fault

When the documentation does not match what the application does, the application has documentation faults. Usually, documentation is derived from system design and provides a clear description of what the programmer would like to program to do, but the implementation of these functions is faulty. Such faults can lead to other faults later.

7.3 Test Planning

The purpose of having test planning is to help in designing and organizing tests, so that testing is carried out appropriately and thoroughly.

A test plan has the following steps:

1. Establishing test objectives

   At the beginning, we have to know what we are going to test on. So we have to establish our test objectives that tell us what kinds of test cases to generate.

2. Designing test cases

   After establishing test objectives, we begin to design the test cases that are used to test the system. If test cases are not representative and do not thoroughly exercise the functions that demonstrate the correctness and validity of the system, then the reminder of the testing process is useless.

3. Writing test cases

   After designing, we have to start writing the test cases.

4. Testing test cases

   At the same time, we also test the test cases.

5. Executing tests

   After all testing have been done, we execute our tests on the system.

6. Evaluating test results

   After executing tests, we evaluate the test results.

7.4 Testing Strategy

Testing is a process of exercising or evaluating a system by manual or automatic means to verify that it has satisfied requirements or to identify differences expected and actual results. Testing is probably the least understood part of a software development project. A bug is any unexpected, questionable, or undesired aspect or behavior displayed, facilitated or caused by the software being tested. Testing can uncover different classes of errors in a minimum amount of time and with a minimum amount of effort. The strategies used for testing are unit testing, module testing, integration testing and system testing.

7.4.1 Unit Testing

Historically, quality software is relied on testing each function or module. Unit testing is sometimes referred as function testing or component testing, which is extremely time-consuming. For this project, unit testing was done during the coding phrase.

The first step is to examine the program code by reading through it, trying to spot algorithm, data and syntax faults. Then, to make sure that all relevant cases have been considered followed by comparing the code with specifications and with the design. Finally, test cases are developed to show that the input is properly converted to the desired output.

Unit testing involves the tests on each function module independently. If error is found, debugging of the codes will be carried out immediately. If the compilation of the function module is completed successfully, another function module will be coded. The following steps specify how unit testing is carried out for this system:

1. **Examining The Code**

   The code of the program is examined by reading through it to spot for algorithmic faults and syntax faults. This method is useful to identify faults that have been left out by the programmer.

2. **Control Objects Testing**

   Command buttons are clicked to test their functionality and text boxes are tested with different data types and also null value to make sure invalid data will not cause any fault.

3. **Choosing Test Cases**

   Test cases are developed to ensure that the input is properly converted to the desired output. So, to test a component, input data and condition are chosen. Then the component is allowed to manipulate the data, and output is observed.

7.4.2 Module Testing

A module is a collection of dependent components. A module encapsulates all of the related components. Module testing enables each module to be tested independently. This testing will ensure that the module calling sequence in this project is systematic. In module testing, two or more units in which either unit that use output data from or provide input data for another unit were tested in collection.

7.4.3 Integration Testing

When the individual components are working correctly and meet the objectives, these components are combined into a working system. In other words, integration testing is the process of verifying that the system components work together as described

in the system and program design specifications. This integration is planned and coordinated so that when a failure occurs, some idea of what caused it can be got.

The motive behind this testing is to make certain that all modules can be executed as a complete module. As mentioned earlier, an individual module calls other module to perform certain tasks. Parameters will be passes among these modules and if not tested, then parameter may be passed incorrectly.

7.4.4 System Testing

The last testing procedure done is system testing. Testing the system is very different from unit testing, module testing and integration testing. The objective of unit testing, module testing and integration testing is to ensure that the code has implemented the design properly. In other words, the code is written to do what the design specifications intended. In system testing, a very different objective is to be achieved, that is to ensure that the system does what the users want it to do.

3D Integration tool is involves two kinds of system testing. They are function testing and performance testing.

7.4.4.1 Function Testing

Function testing is based on the system functional requirements. In other words, a function test is used to check that whether the integrated system performs its functions as specified in the requirements. The testing is carried out for main modules in this system. Each module is tested individually to determine whether the system performs as required.

7.4.4.2 Performance Testing

Performance testing addresses the nonfunctional requirements of the system. That means once the functions are convinced work as specified, the performance test compares the integrated components with the nonfunctional system requirements. The types of performance tests are:

a) Compatibility Tests

This test was performed to find out that the interface functions perform according to the requirements. The accuracy of data retrieval was high in this system.

b) Human Factors Tests

This test was performed to investigate requirements dealing with the user interface to the system. These tests are sometimes called **usability tests**.

c) Recovery Tests

This test was performed to address response to the presence of faults or to the loss of data, power, devices or services.

d) Timing Tests

This test was performed to evaluate the requirements dealing with the time to respond to a user and time to perform a function. The response time of this system is acceptable.

7.5 Summary

Testing is one of the important steps in developing a system. Unit, module, integration and system testing has been carried out for the 3D Integration tool. These testing approaches lead to delivering a quality system to users. The objective of a system will only achieve after all the thorough testing done by different user with different aspects.

Having discussed about the system testing and implementation, the next chapter will see the system evaluation. This coming chapter will touches various things like problems encountered during the development process, system strength and limitation and others.

CHAPTER EIGHT: SYSTEM EVALUATION

Chapter 8: System Evaluation

8.1 Introduction

After having gone through the implementation and testing phase, the final phase of developing this system is the evaluation stage. In this phase, system evaluation involves determining the problems or difficulties, which arise during and after the program coding phase, recognizing the system strengths and weaknesses, and finally draft out the system limitations and also its future enhancements.

8.2 Problem Encountered and Solution

Research and studies on the OpenGL coding and software such as the similar to 3D Integration Tool are not much available in the Internet. There is a specific definition or terms of transformations but it was too general and almost all of the demo systems from the Internet are not in full package and the real system are costly and too complex to develop. At the beginning development of the software, I have confused and misunderstand of the scope of my system. Fortunately, through the guidelines from my supervisor and moderator, I get the idea in developing this software. For this 3D Integration tool, I am focus more on the projecting the 3D environment and transformation. Besides that, grasping the concepts of a totally new programming environment in OpenGL is also takes a lot of my times to learn and master. A study though out the books does not guarantee you know how to apply it practically in your software. Moreover, many problems and difficulties appeared during the whole system development especially in OpenGL coding. Few of the major problems and difficulties encountered during the development and coding process is listed below.

- **Difficulties In Choosing A Development Technology, Programming Language and Tools**

  There are many software tools available to develop 3D software as stated in the earlier chapters. Choosing a suitable technology and tool was a critical process as all tools have their strengths and weaknesses. In addition, the availability of the required tools for development is also a major consideration.

  Solution:

  In order to solve this problem, advises and views were sought from project supervisor, course mates and even seniors engaging in similar project. Furthermore, surfing the Internet and visiting the library helped to clarify some doubts.

- **Difficulties In Programming Language**

  The programming language that been used to develop the whole software is OpengGL. This major programming language being used is totally new for me. Both of us; my partner and me are really having problem in OpenGL especially for save and load function in this project. Hence, these functions are still not available until the completion of this project.

  Solution:

  In order to solve that, seeking help from others thesis students who are also using OpenGL as their development tools and having exchange in views and knowledge helps to gain extra knowledge about the language but there are one student. So it couldn't help me so much. Besides that, by analyzing references source code, and view the output results through the demo, you will get a better understanding of the function writing in those languages. Unfortunately because of time constraints, some functions such as save and import functions are not solved. If I had given longer time and more resources to master OpenGL these problem should be solved.

- **Set Up the Development Environment**

  The operating system needed to run this project is Microsoft Windows XP (SP2). However, Windows ME was preinstalled in the computer that is used for system development. So, the computer has to be formatted. Furthermore, the set up of the library linker between Microsoft Visual C++ 6.0 and Microsoft Visual C++ Dot Net are different. Therefore, the setup process took a long time because lacked of experience. Besides, the repeated failure of the library required re-set up as a remedy and this consumed time and effort.

  Solution:

  This difficulty is solved with the assistance of my friends, those who had experienced to set up such development environment.

- **Problems occur during integration**

  Since the software we handled is a group project, we face most of the problems while doing integration.

  Solution:

  The co-ordination between the group members is mainly important. A lot of details have to be discussed before start developing system especially the OpenGL coding. Some coding needs to be change to integrate part done by me with my partner.

8.3 System Strengths

Although they have been some current software in the market which is similar to this project, but the software are quite straightforward and have limited functions. ***This project proves to be unique in the sense that it can integrate more than one object***. In addition, specific scaling function in this project made 3D objects very precise as output which is rarely been done in most others software.

This project is also specially designed on the principle for ease to use; therefore, all interfaces are kept simple. The inclusion of graphic user interface has contributed vastly to aid users. So, the learning curve is foreseen to be short and a user will be able to use the system with ease within minutes.

8.4 System Constraints and Limitations

- **Failed To Provide Load, Import and Save Function**

  This system didn't provide load, import and save function cause lack of time and knowledge to build it.

- **No Icon for Provided Objects**

  Instead of icon, there are only check boxes for the provided objects.

- **Limited Transformation Function**

  There is transformation function in this software but it is not perfect. Like translation function, it only support for the whole objects involved in the 3D environment. But it is not supported for particular single 3D objects. Same apply for the scaling function in this project. It can't supported to do scaling function on a single 3D objects.

8.5 Future Enhancements

Future enhancement can be done to make the software more advances in order to improve the quality of the system and easier to use. A system development knows no boundaries as new requirements and better implementation methods continue to arise and evolve. There are several enhancements that could extend after developed the system.

- **More Modules and Functions**

  Adding more modules and functions, for example, perspective effect and different viewing angle or direction can be used to attract more users using the system and make the system more interesting.

- **Various Type of Supported Object Files**

  This software should support various types of object files which is common use in the 3D modeling world.

- **More Interactive Interface**

  This software would be more valuable if it can be enhance with more icon and interactive menu and command control. Thus more end user would make this software useful in their works.

8.6 Summary

Evaluation of a system is indeed needed to ensure its objectives and intended functions have been achieved. This chapter covers all the aspect of evaluating application software. At the end of evaluation, comes the conclusion of this thesis project.

8.7 Conclusions

Overall, this project has achieved and fulfilled the objectives and requirements as a 3D integration tool as determined during system analysis. Anyone who needs to create and modify any 3D object provided can use this system. This software can generates or render accurate and precise any 3D objects. The feasibility of the system depends on how much the user will benefit from its implementation.

Apart from applying the knowledge in computer programming, a certain degree of knowledge in art and design are needed to complete this project. This is where extra knowledge is gain and could be useful for future purposes. This includes knowledge in system development, programming, concepts and challenges to develop software alone. Programming using OpenGL and VC++ proved to be a valuable experience. Even though programming skills and techniques are important in development, good software engineering techniques must also be applied. Here, theories and knowledge gained throughout the course of computer science studies like system analysis, design and software engineering were literally put into practice. It gives me a strong foundation to take this project as long as to complete it.

Due to the problems encountered during development, it is sad to mention that the developer of the project was almost behind schedule. But with great dedication and beliefs, the developer managed at the end to complete the project.

Finally, there are much more rooms for improvement in this system, especially in terms of implementing a more satisfactory software. With the first step taken, enhancements could still be made with more features added for future version.

All in all, this thesis has armed me with invaluable knowledge and experience. As a result, I am better prepared to face future challenges in life. There are more things to learn and experience in this fast growing world of information age. One has to constantly update oneself to keep up with the changing technology.

REFERENCES

REFERENCES

Book And Publications

[1] Harvey and Paul Deitel (2000) C How to Program. Prentice Hall, Inc.

[2] Programmer Guide. Microsoft DirectX 8.1 (C++)

[3] Pfleeger, Shari Lawrence. (1998). Software Engineering: Theory and Practice. USA: Prentice hall, Inc.

[4] Summerville, Ian. (1998). Software Engineering. (5th.ed.). Harlow, England: Addison Wesley Publishing Company, Inc.

[5] Robinson, Barbara and Prior, Many (1995). System Analysis Technologies. International Thomson Computer Press.

[6] Kendall & Kendall (1999). System Analysis and Design, 4th Edition. Prentice Hall.

Internet Resources

[7] The Amazing World of Virtual 3D Environments [Online]. Available HTTP: http://www.expresscomputeronline.com/20030630/multi6.shtml
Date Referred: 2004, April 7

[8] Direct3D vs. OpenGL: Which API to Use When, Where, and Why. By Promit Roy. [Online]. Available HTTP:
www.gamedev.net/reference/articles/article1775.asp
Date Referred: 2004, April 2.

[9] Microsoft.com - Web and Application Services (2000). [Online]. Available HTTP: http://www.microsoft.com/windows2000/technologies/web/default.asp Date Referred: 2004, April 5.

[10] Microsoft.com - Extending Your Business with Advanced Web and Application Services (1999). [Online] Available HTTP: http://www.microsoft.com/windows2000/server/evaluation/business/appsvcs.asp #createsite Date Referred: 2004, April 5.

[11] Windows XP Home Edition vs. Professional Edition: What's the difference? February 8, 2001 By Paul Thurrott [Online] Available HTTP: http://www.winsupersite.com/showcase/windowsxp_home_pro.asp Date Referred: 2004, April 5.

[12] Chapter 6: Creating Scene Graphs. Part II. Programming with OpenGL Performer. http://techpubs.sgi.com/library/dynaweb_docs/linux/SGI_Developer/books/Perf_GetStarted/sgi_html/ch06.html Date Referred: 2004, April 7.

[13] Cornell Theory Center Glossary of Visualization Terms [Online]. Available HTTP: http://tc.cornell.edu/services/edu/topics/OpenDX/glossary.html Date Referred: 2004, April 6.

[14] A Layperson's Guide to The Jargon of The Internet [Online]. Available HTTP: http://support.jvlnet.com/etc/glossary.htm. Date Referred: 2004, April 8.

[15] Glossary of Common Internet Terms, Mount Holyoke College [Online]. Available HTTP: http://www.mtholyoke.edu./help/creating-pages/terms.shtml Date Referred: 2004, April 8.

[16] NWIC Technology Glossary [Online]. Available HTTP: http://gslis.utexas.edu/~library/glossary Date Referred: 2004, April 10.

USER MANUAL

3D INTEGRATION TOOL

TABLE OF CONTENTS

List of Figures

Chapter 1: Introduction

This user manual is useful for users from different background to use the 3D integration tool. This user manual consists of the guideline and example to help users to use the software in the correct way. After reading this user manual, the users will be able to use this software in the correct manner to get the very best out of the software.

1.1 About This Manual

This user manual will guide you through all the functions available in this software. This manual includes the following part such as:

- Installation
- Transformation Functions
- Other Features

Chapter 2: Hardware Requirements

Hardware Requirements are stated below:

- Pentium II 800 MHz / Higher microprocessor / or equivalent
- 128MB RAM and above recommended
- 10.5 GB Hard disk and above
- VGA or high-resolution monitor
- Microsoft mouse or compatible pointing device.

Chapter 3: Installation

1. To begin 3D integration tool installation double click its installation executed file.

Figure 3.1 Installation Executed File

2. Then a pop up window will appear to notify you that 3D integration tool will be install to your computer. Click next to continue installation.

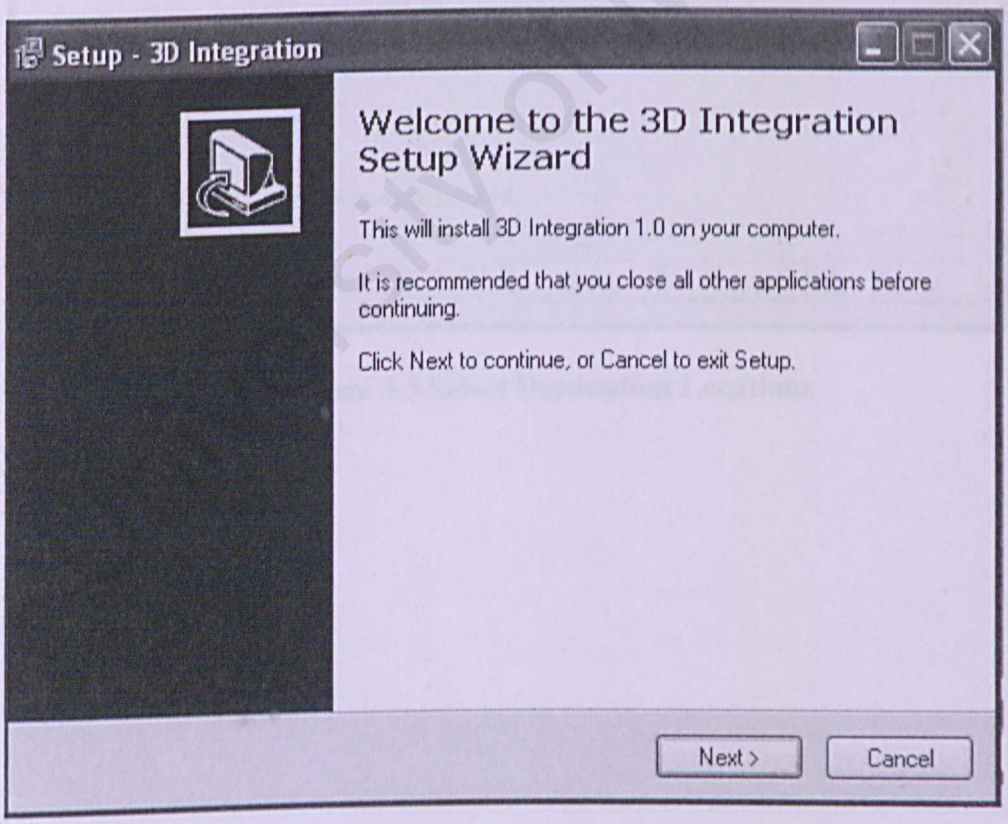


Figure 3.2: Installation Setup

3. Next you will be notified to select the destination folder. Usually just use the default configuration. You are also will notify disk spaces required to install this software. Then click next.

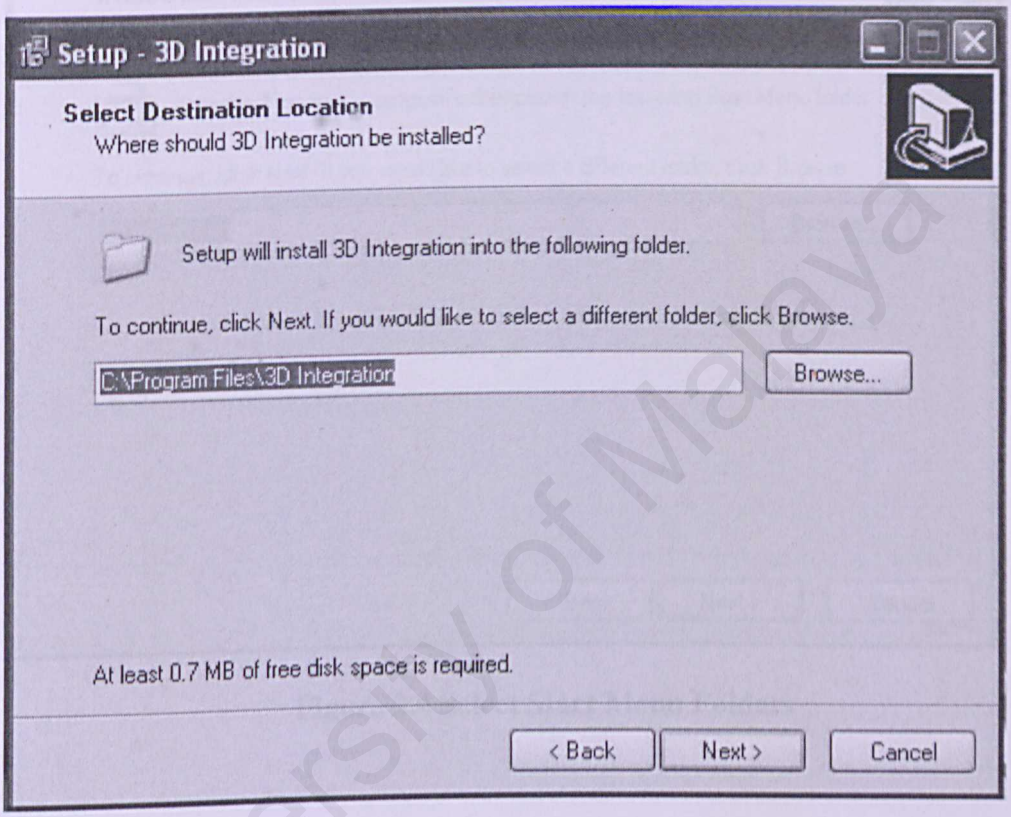


Figure 3.3 Select Destination Locations

4. Then setup will create the program's shortcut to the Start Menu folder. You are recommended to leave it to the default folder. Then click next to continue.

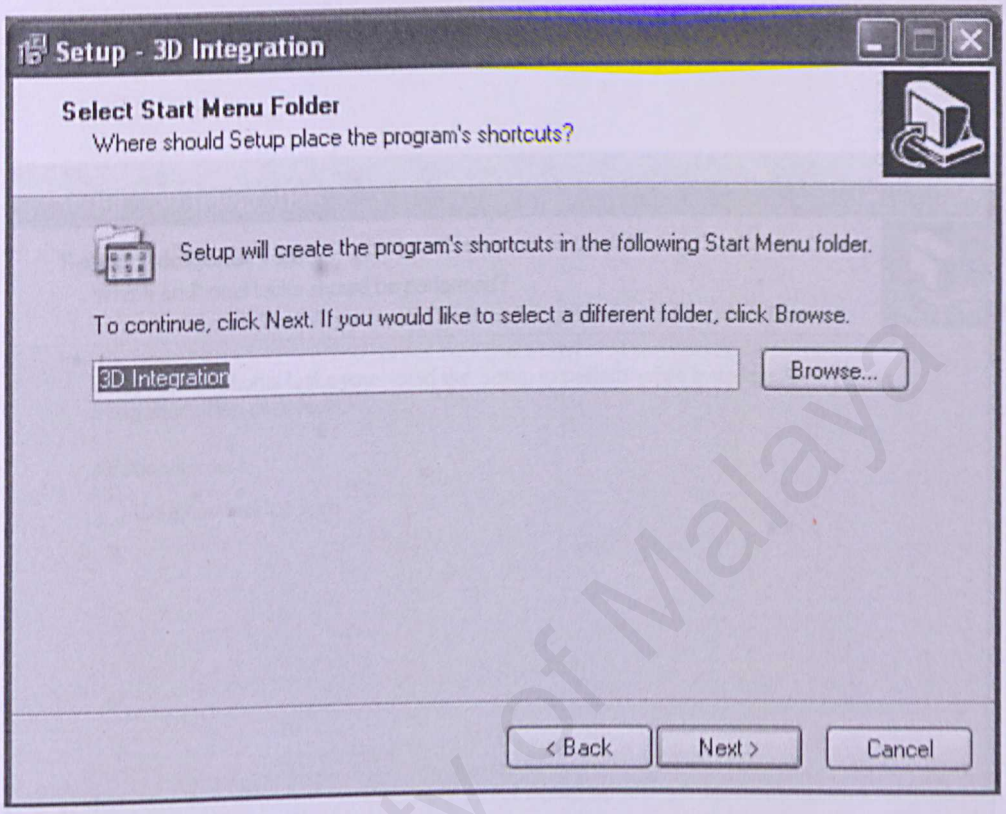


Figure 3.4 Select Start Menu Folders

5. After that setup will offer additional task which is to create a desktop icon. If you prefer a 3D integration tool icon to be placed on your desktop checked the options or else click next to continue.

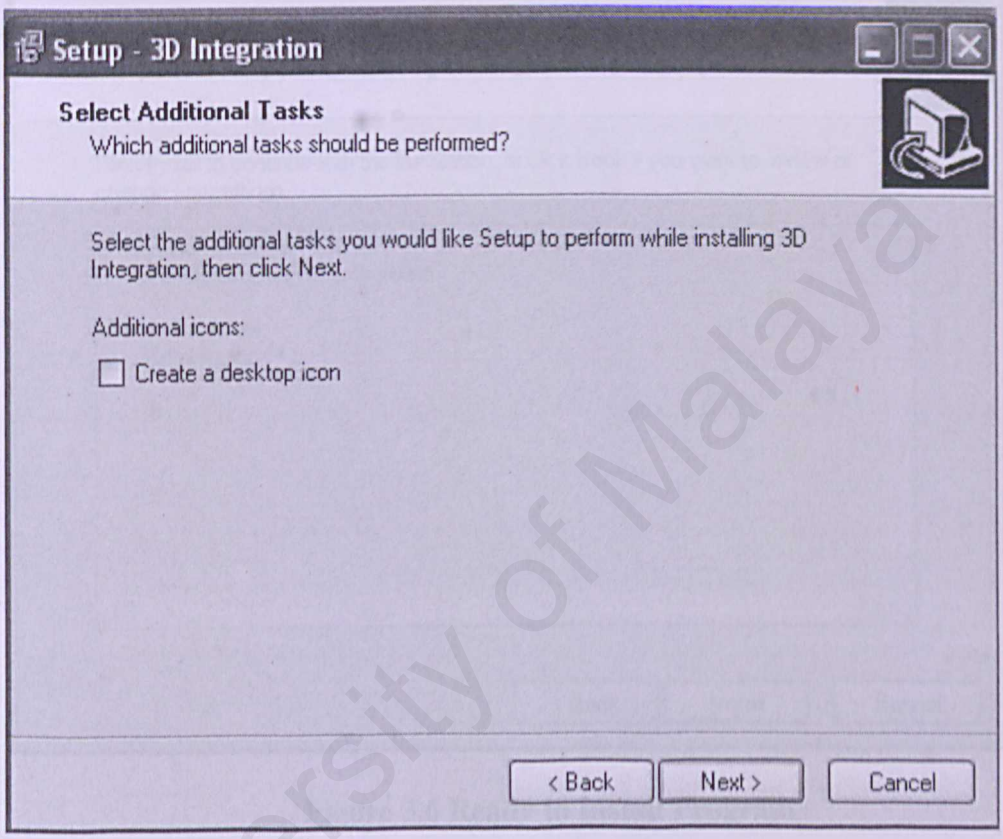


Figure 3.5 Select Additional Tasks

6. Then, setup will show you the setting that you have made and if you want to change it click back button or click install to begin installation.

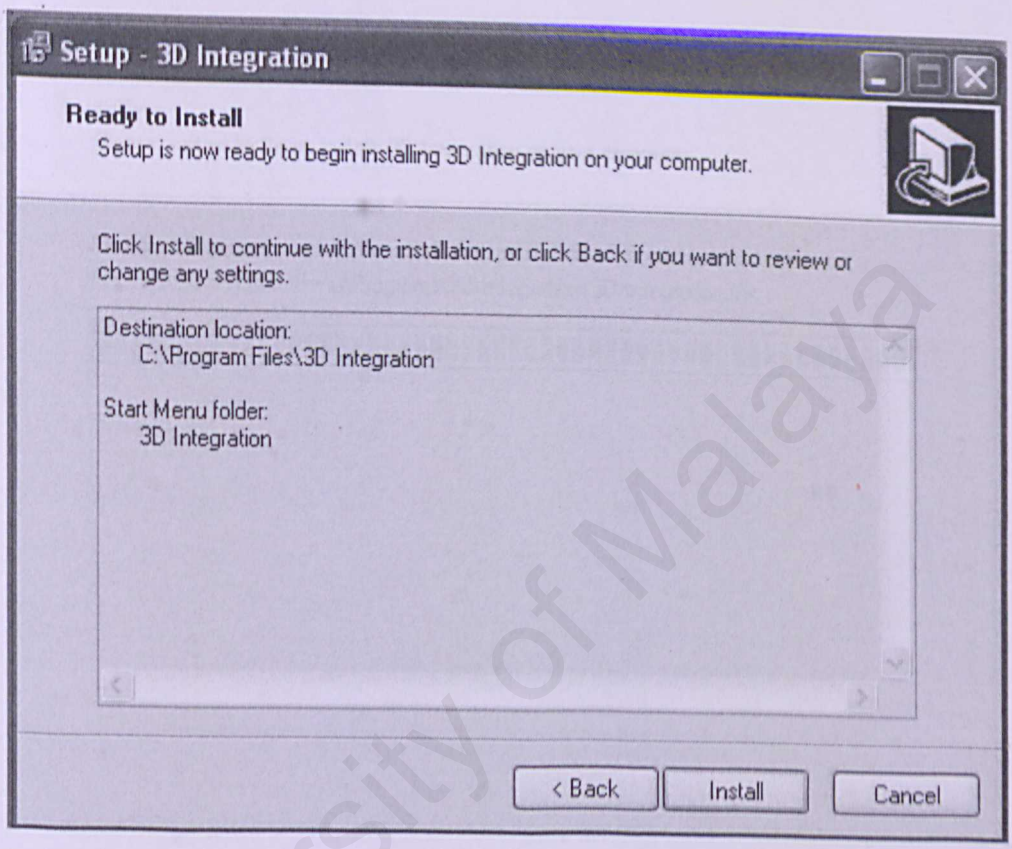


Figure 3.6 Ready to Install Program

7. After you see the installation progress bar complete, then setup will prompt you to launch 3D integration software or click finish to exit the setup.

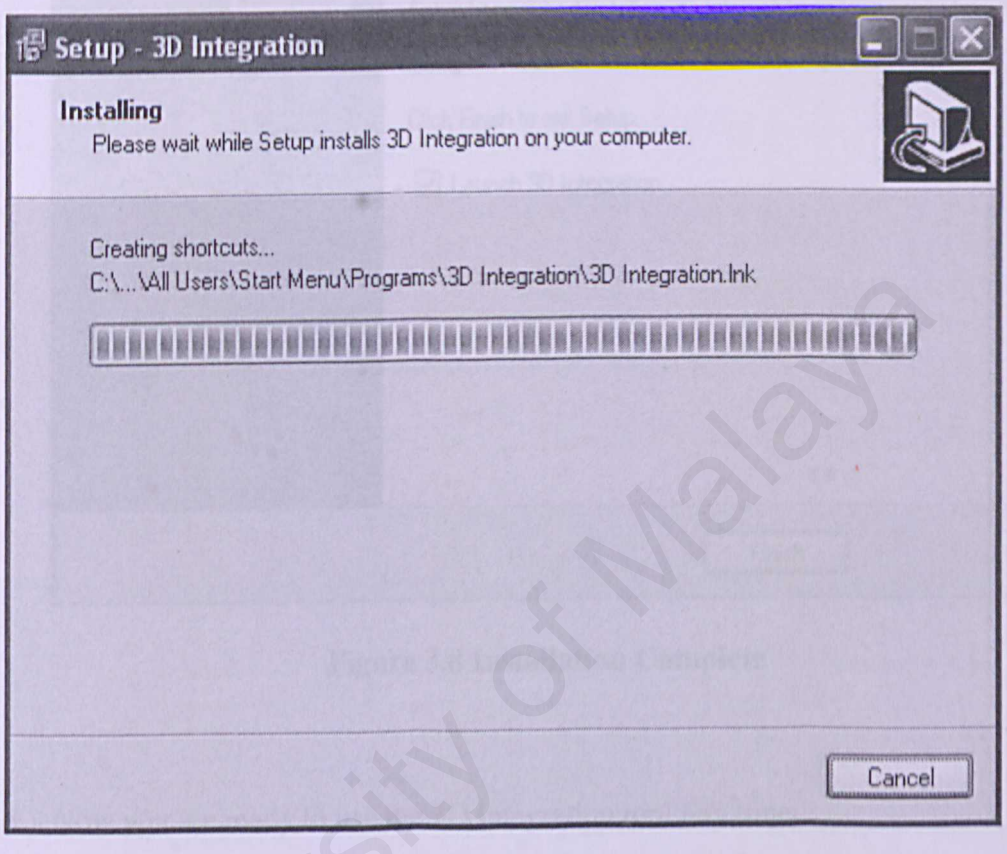


Figure 3.7 Installation Progression Bar

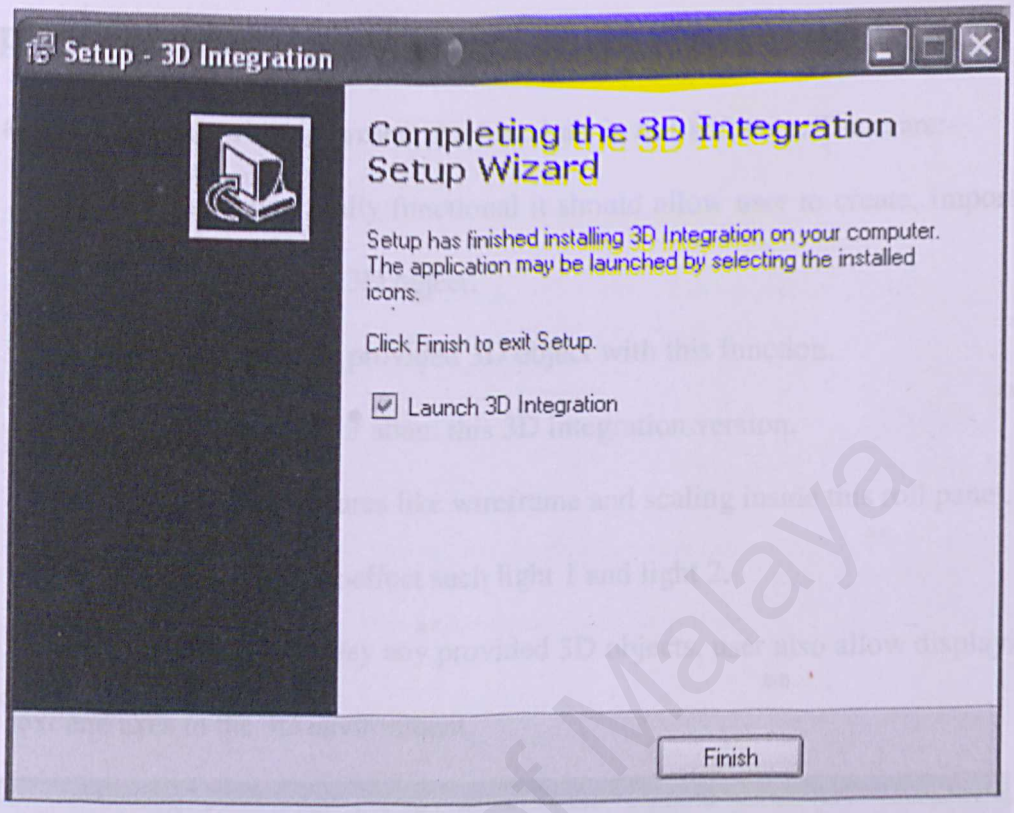


Figure 3.8 Installation Complete

8. Now you are ready to use the 3D integration tool any time.

Chapter 4: Getting Started

There are 6 roll panels, which represent the functions in this software. There are:-

1. Menu – if this feature really functional it should allow user to create, import or load a 3D environment or 3D object.
2. View – user can view any provided 3D object with this function.
3. Help – it's actually show or about this 3D integration version.
4. Properties – there are features like wireframe and scaling inside this roll panel.
5. Lights – user can use light effect such light 1 and light 2.
6. Options – beside can display any provided 3D objects, user also allow displaying text and axes in the 3D environment.

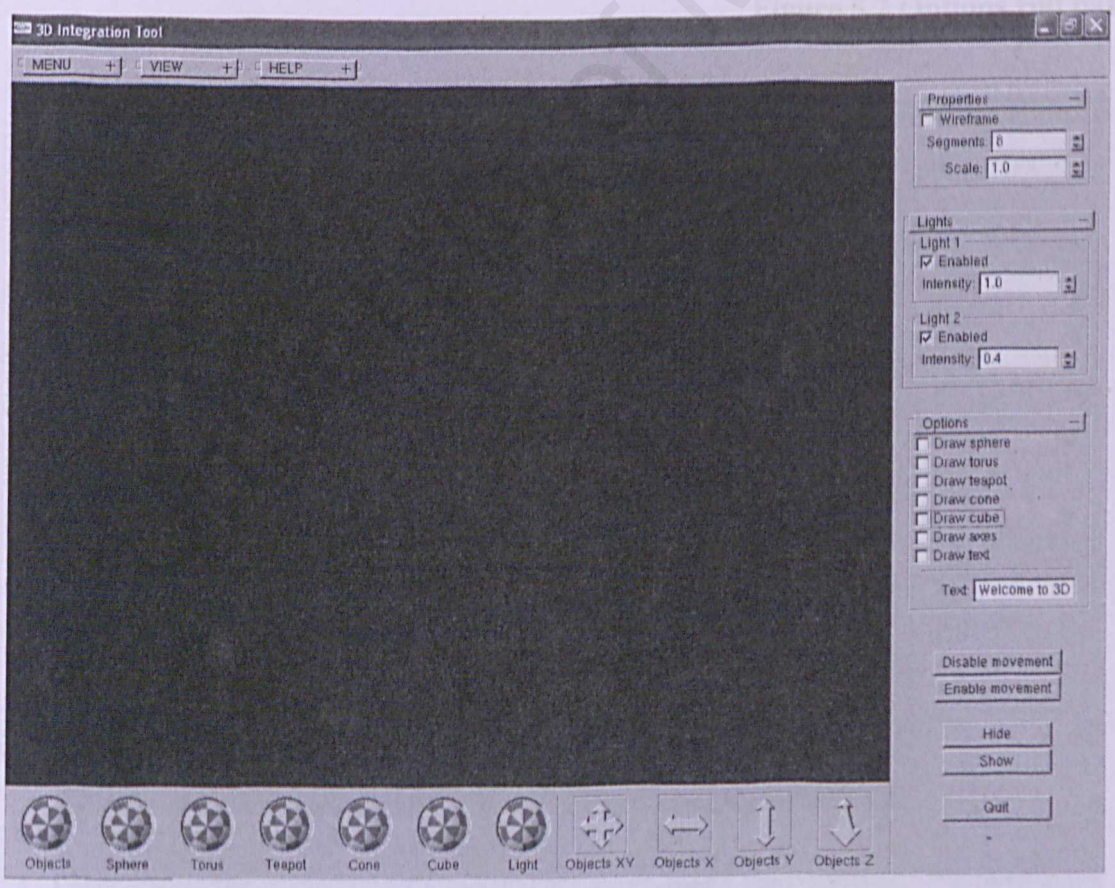


Figure 4.1 3D Integration Tool

Chapter 5: Integration Tool

5.1 Display 3D Objects

There are two ways to display any provided objects. There are:

1. Using View function - you can check any object that u desire such as sphere, torus, teapot, cube or cone.

Figure 5.1 View roll panel

2. Using Options functions – you also can display any 3D provided object with only check the boxes and you also can display axes and text with this functions.

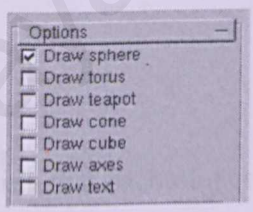


Figure 5.2 Options roll panel

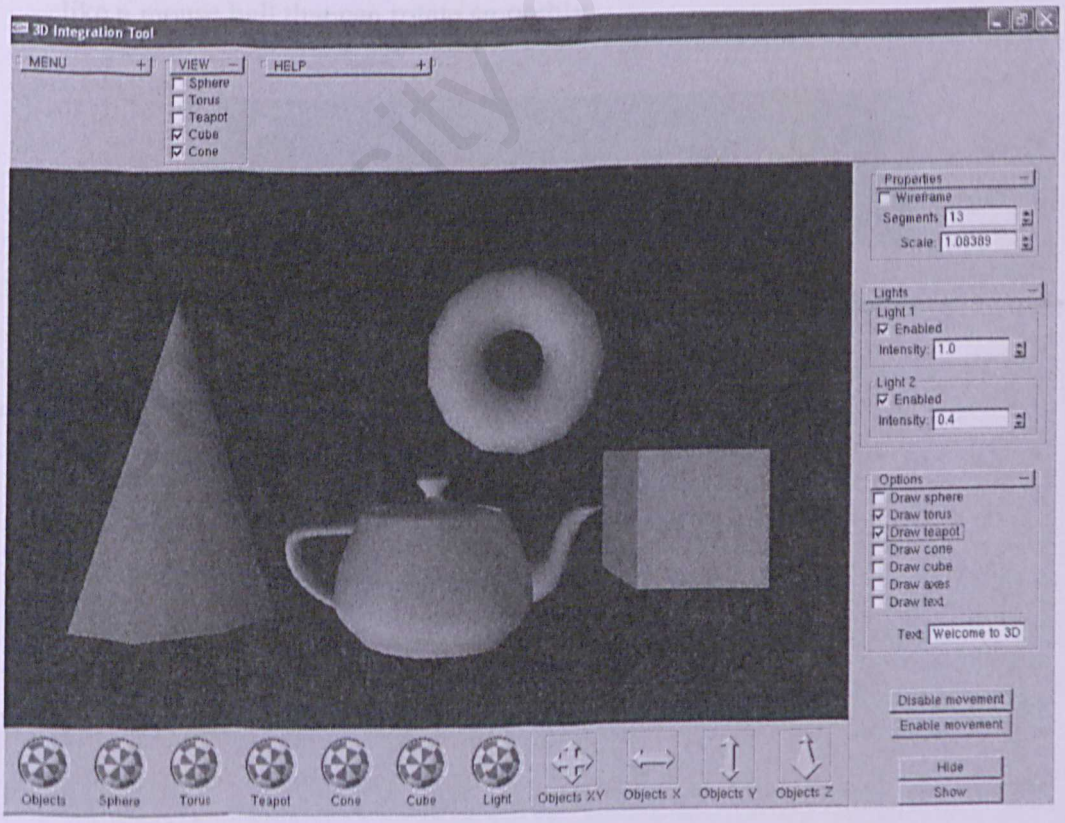


Figure 5.3 Displaying 3D Objects

5.2 Transformations

5.2.1 Rotation

Rotation functions can be done in two ways, which is:

1. Rotation using objects spinner will rotate the whole 3D environment. Meaning that all objects will rotate together using objects spinner.

Figure 5.4 Objects Spinner

2. Rotation also can be done on each 3D object spinner. Meaning that each object can rotate alone with each own spinner. Note that spinner is a mouse sensitive like a mouse ball that can rotate smoothly.

Figure 5.5 Sphere, Torus, Teapot, Cone and Cube Spinner

5.2.2 Translation

There are four arrow buttons for translation function.

1. You can click and drag to translate within both X and Y axis simultaneously.

Figure 5.6 Arrow Button for Translation XY Axis.

2. With this arrow button, you are allow to translate only within X axis only.

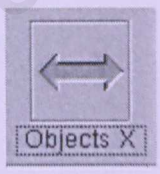


Figure 5.7 Arrow Button for Translation X Axis.

3. This button allows you to translate the whole 3D environment simultaneously within Y axis only.

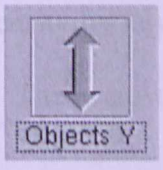


Figure 5.8 Arrow Button for Translation Y Axis.

4. With this arrow button, you are allowed to translate only within Z axis only.

Figure 5.9 Arrow Button for Translation Z Axis.

5.2.3 Scaling

Scaling function can only be done for whole 3D environment. Means that once you key in exact scale that you desire, then all objects will transform according to your scale. This scaling function can be done within 0.2 to 4.0 scales only.

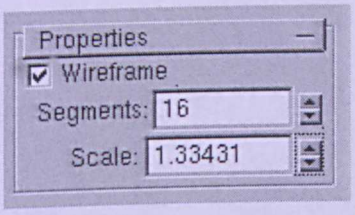


Figure 5.10 Scaling Functions

5.3 Other Features

5.3.1 Lighting

Lighting effect can be done in this tool. You modify lighting intensity by key in exact scale or using the up and down arrow within 0.0 to 0.1 scales. There are two lighting effect; Light 1 and Light 2.

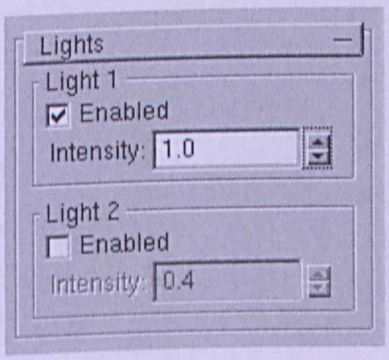


Figure 5.11 Lighting Effect

Figure 5.12 Light 1 and Light 2 are disable.

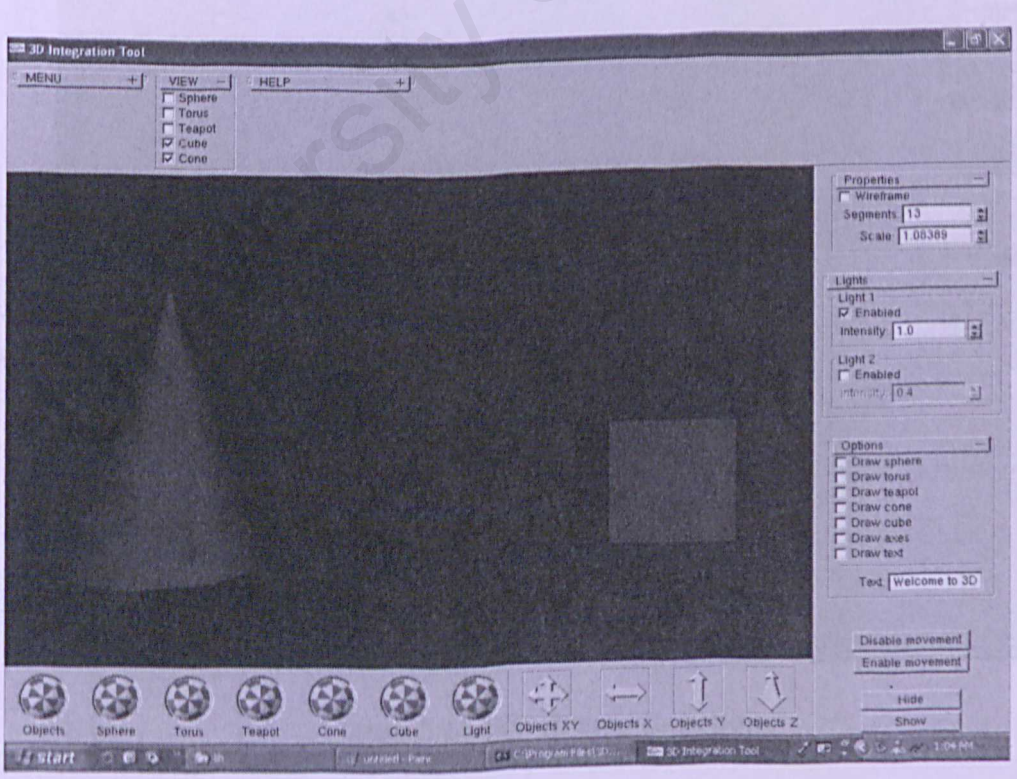


Figure 5.13 Only Light 1 is enable.

5.3.2 Wireframe

Wireframe function can be use to see through segment that lies inside every 3D objects. Segment in certain objects can be manually modified in the properties roll panel. The see the wireframe, checked the wireframe box.

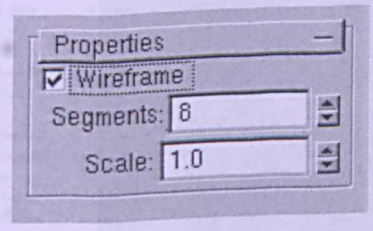


Figure 5.14 Wireframe Roll Panel

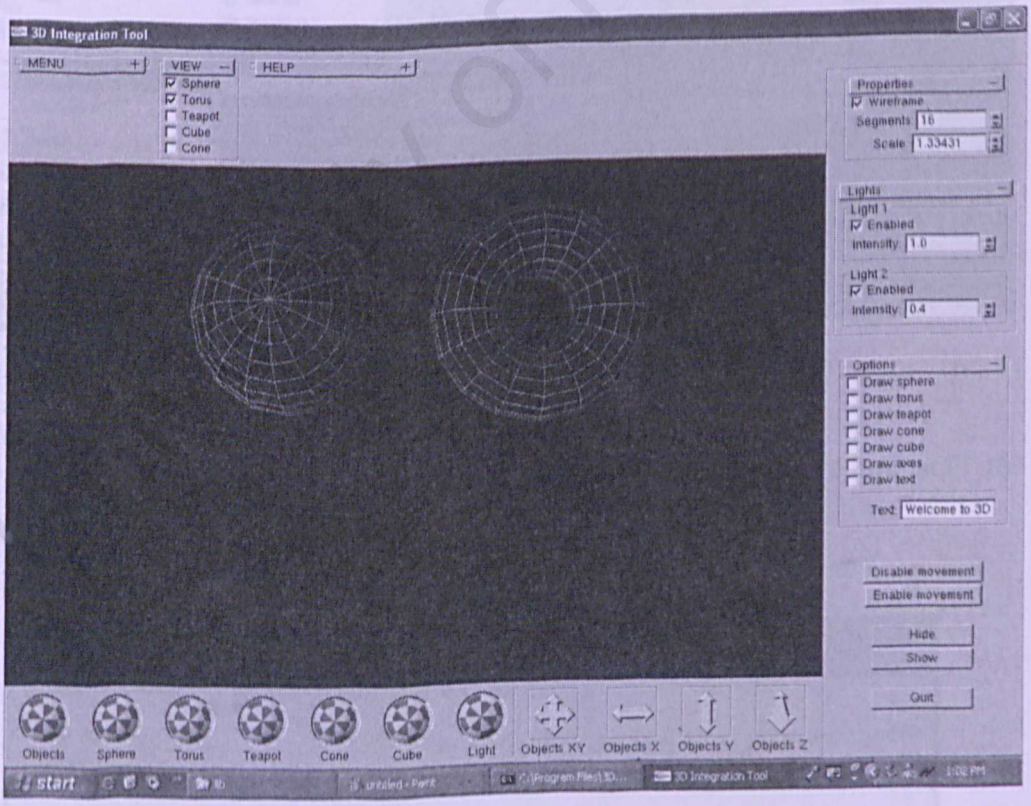


Figure 5.15 Wireframe Functions

Chapter 6: Uninstallation

1. To uninstall 3D integration tool, just click on to its uninstaller from the Start Menu.

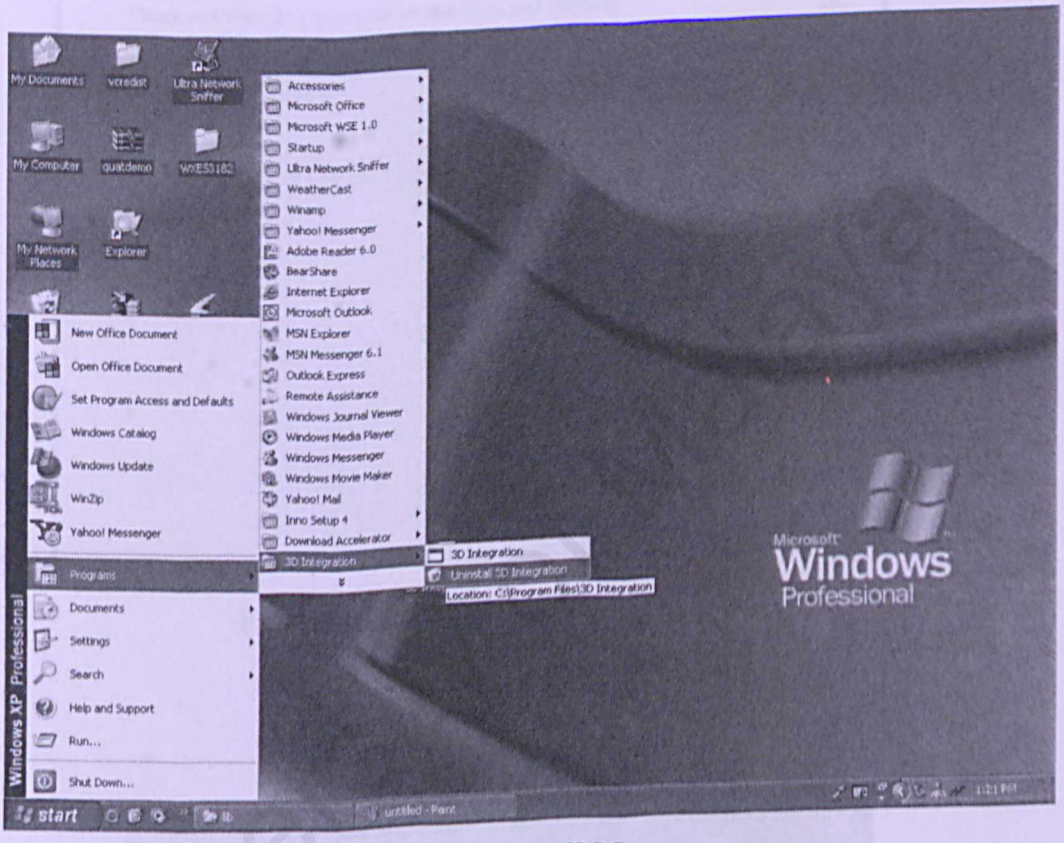


Figure 6.1 Uninstall Menu

2. If you are really sure, click Yes after the setup prompt before uninstall the software.

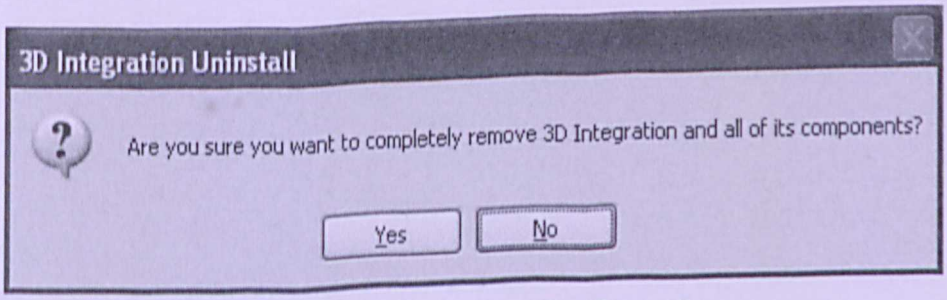


Figure 6.2 Uninstallation Confirmation

3. After finished uninstall, another pop up messages notify you that uninstall are complete.

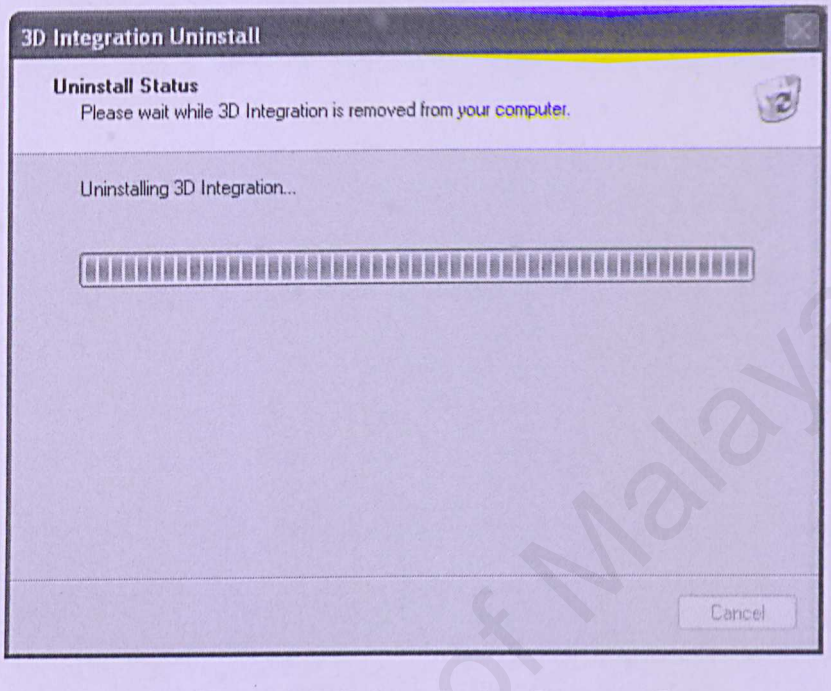


Figure 6.3 Uninstallation Progress Bar

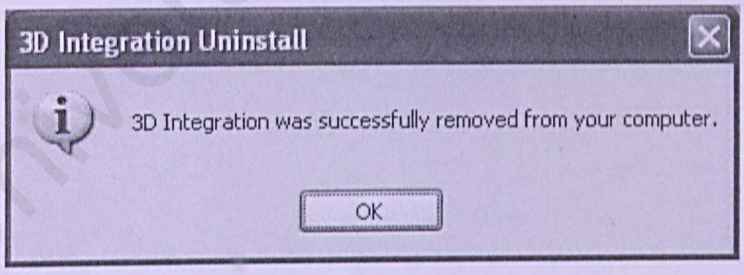


Figure 6.4 Uninstallation Complete